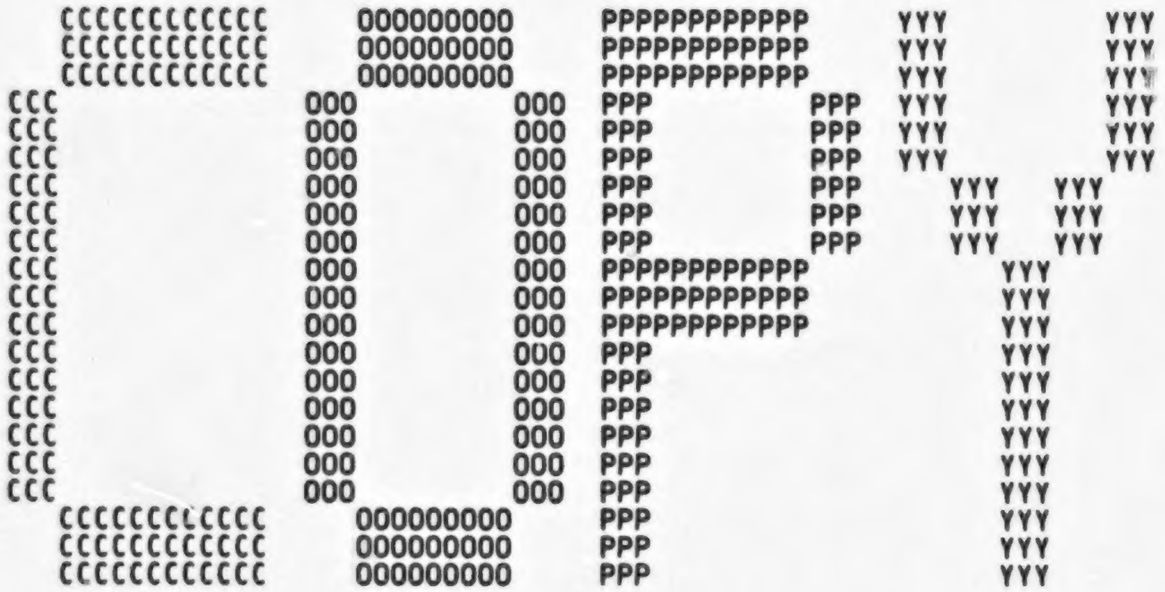
CCCCCCCCCCC OOOOOOOOO PPPPPPPPPPPP YYY YYY
CCCCCCCCCCC OOOOOOOOO PPPPPPPPPPPP YYY YYY
CCCCCCCCCCC OOOOOOOOO PPPPPPPPPPPP YYY YYY
CCC OOO OOO PPP PPP YYY YYY
CCC OOO OOO PPP PPP YYY YYY
CCC OOO OOO PPP PPP YYY YYY
CCC OOO OOO PPP PPP YYY YYY
CCC OOO OOO PPP PPP YYY YYY
CCC OOO OOO PPP PPP YYY YYY
CCC OOO OOO PPPPPPPPPPPP YYY
CCC OOO OOO PPPPPPPPPPPP YYY
CCC OOO OOO PPPPPPPPPPPP YYY
CCC OOO OOO PPP YYY
CCC OOO OOO PPP YYY
CCC OOO OOO PPP YYY
CCC OOO OOO PPP YYY
CCC OOO OOO PPP YYY
CCC OOO OOO PPP YYY
CCCCCCCCCCC OOOOOOOOO PPP YYY
CCCCCCCCCCC OOOOOOOOO PPP YYY
CCCCCCCCCCC OOOOOOOOO PPP YYY

FILEID**COPYMAIN

```
  CCCCCCCC    000000    PPPPPPPP    YY      YY  MM      MM    AAAAAA      IIIIII    NN      NN
  CCCCCCCC    000000    PPPPPPPP    YY      YY  MM      MM    AAAAAA      IIIIII    NN      NN
CC          00      00  PP      PP  YY      YY  MMMM  MMMM  AA      AA      II      NN      NN
CC          00      00  PP      PP  YY      YY  MMMM  MMMM  AA      AA      II      NN      NN
CC          00      00  PP      PP    YY  YY    MM  MM  MM  AA      AA      II      NNNN    NN
CC          00      00  PP      PP    YY  YY    MM  MM  MM  AA      AA      II      NNNN    NN
CC          00      00  PPPPPPPP        YY      MM      MM  AA      AA      II      NN  NN  NN
CC          00      00  PPPPPPPP        YY      MM      MM  AA      AA      II      NN  NN  NN
CC          00      00  PP              YY      MM      MM  AAAAAAAAAA      II      NN    NNNN
CC          00      00  PP              YY      MM      MM  AAAAAAAAAA      II      NN    NNNN
CC          00      00  PP              YY      MM      MM  AA      AA      II      NN      NN      ....
CC          00      00  PP              YY      MM      MM  AA      AA      II      NN      NN      ....
  CCCCCCCC    000000    PP              YY      MM      MM  AA      AA    IIIIII    NN      NN      ....
  CCCCCCCC    000000    PP              YY      MM      MM  AA      AA    IIIIII    NN      NN      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE COPYMAIN (IDENT = 'V04-000',
;     2          0002  0                   MAIN = COPY$COPY
;     3          0003  0                   ) =
;     4          0004  1 BEGIN
;     5          0005  1
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.               *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                 *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:  COPY
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This utility program creates a copy of one or more user-specified
;    36          0036  1 !       files.  Two or more files may optionally be concatenated to
;    37          0037  1 !       create a single output file.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:
;    40          0040  1 !
;    41          0041  1 ! AUTHOR:  Ward Clark,  CREATION DATE:  19 August 1977
;    42          0042  1 !
;    43          0043  1 ! Modified by:
;    44          0044  1 !
;    45          0045  1 !       V03-014 TSK0015         Tamar Krichevsky          26-Jul-1984
;    46          0046  1 !               Use the constant 32 for the multi-block count, instead of
;    47          0047  1 !               the system multi-block count.
;    48          0048  1 !
;    49          0049  1 !       V03-013 TSK0014         Tamar Krichevsky          9-jun-1984
;    50          0050  1 !               Avoid an access violation by have BYPASS_CONCAT return a value.
;    51          0051  1 !               If this value is true, then stop processing.  If it is false,
;    52          0052  1 !               then continue copying files.
;    53          0053  1 !
;    54          0054  1 !       V03-012 TSK0013         Tamar Krichevsky          8-May-1984
;    55          0055  1 !               Rearrange the calls to CLI$GET_VALUE and LIB$FIND_FILE so that
;    56          0056  1 !               a command such as COPY a.a,a.a,a.a,a.a NL: will copy every file,
;    57          0057  1 !               instead of every other file.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-011 TSK0012         Tamar Krichevsky        25-Apr-1984
;    60          0060  1 !               Add a check, after trying to open the output file, to be
;    61          0061  1 !               sure that if the current operation is an APPEND and the output
;    62          0062  1 !               file was not found, then processing should stop.  No use
;    63          0063  1 !               appending to a non-existant file.
;    64          0064  1 !
;    65          0065  1 !       V03-010 TSK0011         Tamar Krichevsky        17-Mar-1984
;    66          0066  1 !               Add a missing ".", so that the correct files are opened when
;    67          0067  1 !               the input file has a wildcard in it's specification.  Copy the
;    68          0068  1 !               resultant file file name from LIB$FIND_FILE into the input
;    69          0069  1 !               file's NAM block and IN_NAME_DESC.  Otherwise, the confirm
;    70          0070  1 !               prompt, log messages and error reporting would use the wrong
;    71          0071  1 !               information.
;    72          0072  1 !
;    73          0073  1 !       V03-009 TSK0010         Tamar Krichevsky        27-Feb-1984
;    74          0074  1 !               Replace COPY's scheme for allocating I/O buffer pool (The I/O
;    75          0075  1 !               buffer pool is area in which COPY maintains its user buffers
;    76          0076  1 !               for RMS calls.)  The old scheme allocated virtual memory for
;    77          0077  1 !               the I/O buffer pool based on the processes working set size.
;    78          0078  1 !               The new scheme alloates enough virtual memory to hold the
;    79          0079  1 !               largest record or block transfer instead.
;    80          0080  1 !
;    81          0081  1 !               Convert input file parse and searching to LIB$FIND_FILE.
;    82          0082  1 !
;    83          0083  1 !       V03-008 TSK0009         Tamar Krichevsky        15-Feb-1984
;    84          0084  1 !               Fix RMS_SETUP so that the incompatible attributes message is
;    85          0085  1 !               not issued when the input or the output device is network.
;    86          0086  1 !
;    87          0087  1 !       V03-007 TSK0008         Tamar Krichevsky        3-Oct-1983
;    88          0088  1 !               Fix RMS_SETUP so that the incompatible attributes message is
;    89          0089  1 !               not issued when the input device is a unit record device.
;    90          0090  1 !               The input and output devices have to be the same kind of devices
;    91          0091  1 !               and be file structured before the information in the file header
;    92          0092  1 !               can be compared.
;    93          0093  1 !
;    94          0094  1 !       V03-006 TSK0007         Tamar Krichevsky        6-Sep-1983
;    95          0095  1 !               Fix an Access violation introduced in V30-005.  This time
;    96          0096  1 !               wild card copy operations didn't work.
;    97          0097  1 !
;    98          0098  1 !       V03-005 TSK0006         Tamar Krichevsky        1-Sep-1983
;    99          0099  1 !               Fix access violation introduced in V30-004.  Append operations
;   100          0100  1 !               didn't work.
;   101          0101  1 !
;   102          0102  1 !       V03-004 TSK0005         Tamar Krichevsky        29-Aug-1983
;   103          0103  1 !               Modify how the output file's XAB chain is reinitialized at the
;   104          0104  1 !               end of COPY$COPY.  This change has been made so that COPY
;   105          0105  1 !               adheres to the new philosophy about the propogation of
;   106          0106  1 !               file protection and revision dates.
;   107          0107  1 !
;   108          0108  1 !       V03-003 TSK0004         Tamar Krichevsky        23-Jan-1983
;   109          0109  1 !               Replace the command language interface with the the new CLI.
;   110          0110  1 !
;   111          0111  1 !               Add COPY$CHECK_FILE_FOR_MATCH routine which calls LIB$QUAL_FILE_MATCH
;   112          0112  1 !               to see if the input file should be copied to the output file.
;   113          0113  1 !
;   114          0114  1 !       V03-003 TSK0003         Tamar Krichevsky        29-Mar-1982
```

```
;  115          0115  1 ;                Allow /NOTRUNCATE to work for non-contiguous sequential files
;  116          0116  1 ;                by correcting the IF statement in COPY$CALC_ALQ which decides
;  117          0117  1 ;                if the output file will be truncated or the same size as
;  118          0118  1 ;                the input file.  Previously, non-contiguous sequential files
;  119          0119  1 ;                were always being truncated, even if /NOTRUNCATE was specified.
;  120          0120  1 ;                Now, if /NOTRUNCATE is given, the allocation of the input file
;  121          0121  1 ;                is used for the output file.
;  122          0122  1 ;
;  123          0123  1 ;
;  124          0124  1 ;        V03-002 TSK0002         Tamar Krichevsky        22-Mar-1982
;  125          0125  1 ;                Correct logic in IF statement which forces record mode I/O
;  126          0126  1 ;                in RMS_SETUP.  Record mode copies to a foreign disk were being
;  127          0127  1 ;                attempted instead of block mode.
;  128          0128  1 ;
;  129          0129  1 ;        V03-001 TSK0001         Tamar Krichevsky        16-Mar-1982
;  130          0130  1 ;                Force record mode operations if input and output devices are
;  131          0131  1 ;                both magtape and one is ANSI while the other is mounted
;  132          0132  1 ;                foreign.
;  133          0133  1 ;
;  134          0134  1 ;        V021    WMC032          Wayne Cardoza   22-Dec-1981
;  135          0135  1 ;                Don't allow copy of a directory as a file.
;  136          0136  1 ;                Let the [] be displayed in mag tape log messages.
;  137          0137  1 ;
;  138          0138  1 ;        V020    WMC026          Wayne Cardoza   10-Dec-1981
;  139          0139  1 ;                Fix incorrect ordering of PARSE.
;  140          0140  1 ;                Fix log messages for network devices.
;  141          0141  1 ;
;  142          0142  1 ;        V019    WMC003          Wayne Cardoza   17-Nov-1981
;  143          0143  1 ;                Quit when operator aborts a mount request.
;  144          0144  1 ;
;  145          0145  1 ;        V018    WMC002          Wayne Cardoza   02-Nov-1981
;  146          0146  1 ;                Don't try to create directories on record devices.
;  147          0147  1 ;                Make sure directory created in correct directory.
;  148          0148  1 ;                Don't print directory name for non-directory devices.
;  149          0149  1 ;
;  150          0150  1 ;        V017    TMH0017         Tim Halvorsen   06-Sep-1981
;  151          0151  1 ;                Do not issue "N files created" if the number of files
;  152          0152  1 ;                created is only one.
;  153          0153  1 ;
;  154          0154  1 ;        X0016   KRM0007         Karl Malik      11-Feb-1981
;  155          0155  1 ;                Modified COPY$COPY to not attempt to create a directory
;  156          0156  1 ;                when the output is a network device. Instead, issue a
;  157          0157  1 ;                MSG$_NOTCREDIR (new) warning message and continue.
;  158          0158  1 ;
;  159          0159  1 ;        X0015   KRM0005         Karl Malik      14-Jan-1981
;  160          0160  1 ;                Init the block_count and record_count in CREATE_DIR so as
;  161          0161  1 ;                not to use the previous value. Also, modified REPORT_NAMES
;  162          0162  1 ;                to issue a "created" message when a subdirectory is created
;  163          0163  1 ;                (rather than a "copied" message).
;  164          0164  1 ;
;  165          0165  1 ;        X0014   LMK0001         Len Kawell      27-Mar-1980
;  166          0166  1 ;                Correct computation of USZ and MBC for record mode.
;  167          0167  1 ;
;  168          0168  1 ;        X0013   TMH0012         Tim Halvorsen   31-Jan-1980
;  169          0169  1 ;                Do not use LRL as the USZ for record mode I/O as the LRL
;  170          0170  1 ;                can sometimes be incorrect when appending files together
;  171          0171  1 ;                with differing LRL's.  COPY should be fixed sometime in
```

```
;  172          0172  1 !                    the future to make the LRL on a concatenated file correct.
;  173          0173  1 !
;  174          0174  1 !           X0012    JAK0012          J. Krycka       07-Dec-1979
;  175          0175  1 !                    Set ASY bit in ROP after $CONNECT when doing block I/O to
;  176          0176  1 !                    avoid having to issue a $WAIT after the connect. This is
;  177          0177  1 !                    necessary for network block I/O because a network $CONNECT
;  178          0178  1 !                    actually causes DAP messages to be exchanged and thus does not
;  179          0179  1 !                    complete immediately.
;  180          0180  1 !
;  181          0181  1 !           X00011   TMH0011          T. Halvorsen    19-Dec-1979
;  182          0182  1 !                    Do not create a directory on the output side for magtapes.
;  183          0183  1 !
;  184          0184  1 !           X00010   TMH0010          T. Halvorsen    17-Nov-1979
;  185          0185  1 !                    Add GLOBAL ROUTINE msg_number from its own module to
;  186          0186  1 !                    this module to avoid conflict with require file of the
;  187          0187  1 !                    same name in the update procedure.
;  188          0188  1 !                    It had one modification:
;  189          0189  1 !                                    T. Halvorsen    15-Nov-1979
;  190          0190  1 !                    Do not add in COPY/APPEND facility unless high-order
;  191          0191  1 !                    word is non-zero.
;  192          0192  1 !
;  193          0193  1 !           X00009   TMH0009          T. Halvorsen    24-Oct-1979
;  194          0194  1 !                    If input file is a directory file, then either create
;  195          0195  1 !                    a directory on the output side or do nothing depending
;  196          0196  1 !                    on whether the directory already exists or not.
;  197          0197  1 !
;  198          0198  1 !           X00008   T. Halvorsen    16-Aug-1979
;  199          0199  1 !                    Move fixed_overhead to here from COPY.REQ and increase
;  200          0200  1 !                    it by another 10 to avoid copy from magtape wsl problems
;  201          0201  1 !
;  202          0202  1 !           X00007   T. Halvorsen    30-Jul-1979
;  203          0203  1 !                    Make RMS_SETUP fill the UBF/USZ fields for all device types
;  204          0204  1 !                    due to a change in RMS which causes move mode to always be
;  205          0205  1 !                    used (locate mode had some timing windows).
;  206          0206  1 !
;  207          0207  1 !           X00006   T. Halvorsen    21-Jul-1979
;  208          0208  1 !                    Remove 60 second timeout from input RAB
;  209          0209  1 !
;  210          0210  1 !           X00005   T. Halvorsen    14-Jul-1979
;  211          0211  1 !                    Detect insufficient working set size to avoid "internal logic
;  212          0212  1 !                    error" message when allocating negative amount of storage.
;  213          0213  1 !
;  214          0214  1 !           X00004   JAK0004          J. Krycka       16-Mar-1978     15:00
;  215          0215  1 !                    To support file append over the network, omit 'incompatible
;  216          0216  1 !                    attributes' check if NET bit is set.
;  217          0217  1 !
;  218          0218  1 !           X00003   JAK0003          J. Krycka       16-Mar-1978     14:30
;  219          0219  1 !                    To support copy of files in VFC format over the network,
;  220          0220  1 !                    put RHB address in both input and output RABs if NET bit is set.
;  221          0221  1 !
;  222          0222  1 !
;  223          0223  1 ! 01        18-04-78        C. Peters       Change INCLUDE file declarations to suit VMS native compiles.
;  224          0224  1 !                                           Remove SHR$_HASHCONCAT, SHR$_INCOMPAT literals.
;  225          0225  1 ! 02        18-04-78        C. Peters       Change COPY to reflect modified behavior.
;  226          0226  1 !           Include COPY.REQ. Delete LITERAL definitions for general use, status flags. Delete
;  227          0227  1 !                   macro definitions for commonly used status flags.
;  228          0228  1 !           Rename COPY_STATUS to COPY$CLI_STATUS.
```

```
;  229          0229  1 !       Don't include RMSMAC.L32, STARDE.L32. Include STARLET.L32 from SYS$LIBRARY.
;  230          0230  1 !             Delete external literal declarations of RMS status codes. They are in STARLET.L32 too.
;  231          0231  1 !       Delete GLOBAL variable COPY$CLI_STATUS. Put it in a new module, COPYGBL.B32.
;  232          0232  1 !       Instead of calling GET_OUTFILE, call COPY$GET_OUTFIL, in COPYSPECS.
;  233          0233  1 !       Delete GET_OUTFILE.
;  234          0234  1 !       Instead of calling GET_INFILE, call COPY$GET_INFILE, in COPYSPECS.B32.
;  235          0235  1 !       Delete GET_INFILE from this module.
;  236          0236  1 !       Instead of calling OPEN_INFILE, call COPY$OPN_INFILE, in COPYSPECS.
;  237          0237  1 !       Delete OPEN_INFILE.
;  238          0238  1 !       Rename IN_OPEN_ERROR to COPY$INOPN_ERR; OUT_OPEN_ERROR to COPY$OUTOPN_ERR;
;  239          0239  1 !             CLOSE_OUTFILE to COPY$CLOSE_OUTF.
;  240          0240  1 !       Instead of calling OPEN_OUTFILE, call COPY$OPN_OUTFIL, in COPYSPECS.
;  241          0241  1 !       Rename OUT_CLOSE_ERROR to COPY$OCLOSE_ERR.
;  242          0242  1 !       Remove declaration for STS$K_INFO. Put this in COPY.REQ.
;  243          0243  1 !       Remove declaration for VMSMAC.L32, put it in COPY.REQ.
;  244          0244  1 !       Delete routine OPEN_OUTFILE. This routine is replaced by COPY$OPN_OUTFIL, in COPYSPECS.
;  245          0245  1 !       Rename CALCULATE_ALQ to COPY$CALC_ALQ and make it a global routine.
;  246          0246  1 !       Rename MESSAGE_NUMBER to COPY$MSG_NUMBER and make it a global routine.
;  247          0247  1 !       Rename CLI_RESULT to COPY$CLI_RESULT. Declare it a global in COPYGBL.
;  248          0248  1 !       In main routine, close output file is flag MULTIPLE_OUTPUT is set, instead of testing
;  249          0249  1 !             for the CONCAT_FOLLOWS flag being not set.
;  250          0250  1 !       Move setting of CONCAT_QUAL and NOCONCAT_QUAL into the routine GET_CMD_QUAL.
;  251          0251  1 !       Move OUTFILE_OPEN and APPEND_COMMAND bits into COPY$SEM_STATUS from COPY$CLI_STATUS.
;  252          0252  1 !       Remove RMS declarations for input file descriptions to file called FILINPUT.B32.
;  253          0253  1 !       Remove RMS declarations for output file descriptions to file called FILOUTPUT.B32.
;  254          0254  1 !       Rename PARSE_INFILE to COPY$PARS_INFIL.
;  255          0255  1 !       Move PUT_MESSAGE and PUT_MESSAGEX macro definitions to include file COPYMSG.REQ.
;  256          0256  1 !       Move routine COPY$MSG_NUMBER to new module, COPYMSG.B32.
;  257          0257  1 !       In CALC_ALQ, if /TRUNCATE was specified without /ALLOCATION, calculate allocation
;  258          0258  1 !             value based on actual EOF of input file.
;  259          0259  1 !       Add a global variable COPY$B_INCOMPAT. If this variable is set, don't output
;  260          0260  1 !             incompatible attributes message because it has already been output once
;  261          0261  1 !             for this output file.
;  262          0262  1 !       In RMS_SETUP, when setting the MBC and MBF fields for a record mode copy,
;  263          0263  1 !             set the MBC field to the size of the input file only the size is less than or
;  264          0264  1 !             equal to 127 blocks. Otherwise, MBC goes negative.
;  265          0265  1 !       In RMS_SETUP, a record mode copy from disk or tape loads RAB$W_USZ from XAB$W_LRL if
;  266          0266  1 !             non-zero; otherwise, FAB$W_BLS.
;  267          0267  1 !
;  268          0268  1 !--
```

```
;  270          0269  1 !++
;  271          0270  1 ! DETAILED FUNCTIONAL DESCRIPTION:
;  272          0271  1 !
;  273          0272  1 !       This utility program creates a copy of one or more user-specified
;  274          0273  1 !       files.  These files can be explicitly named or can be referred to
;  275          0274  1 !       through use of RMS wildcard file naming.  Two or more files may
;  276          0275  1 !       optionally be concatenated to create a single output file.
;  277          0276  1 !
;  278          0277  1 !       All file I/O is done using standard RMS facilities.  Therefore,
;  279          0278  1 !       the input and output files can exist on any device supported by RMS,
;  280          0279  1 !       including devices at remote network nodes.  If possible, file copying
;  281          0280  1 !       is done using block I/O.  Record I/O is used only when an input or
;  282          0281  1 !       output file is record oriented (e.g., terminal, unit record) or when
;  283          0282  1 !       a concatenated file is being copied.
;  284          0283  1 !
;  285          0284  1 !       This utility is intended to interface directly with a Command Language
;  286          0285  1 !       Interpreter (CLI) and cannot be directly invoked from Command Language
;  287          0286  1 !       level or from an executing program.  Numerous command options
;  288          0287  1 !       (i.e., qualifiers) are supported to allow the Command Language user
;  289          0288  1 !       to (1) optionally specify the location and attributes of the input
;  290          0289  1 !       and output files, and (2) control the reporting of each file copy.
;  291          0290  1 !
;  292          0291  1 !       If more than one copy operation is specified in a single COPY request,
;  293          0292  1 !       each file copy is performed independent of the others.  Therefore,
;  294          0293  1 !       the failure of one file copy operation (e.g., I/O error, input
;  295          0294  1 !       file not found) does not affect the remaining copy requests.  The
;  296          0295  1 !       single exception to this rule is that unprocessed concatenated input
;  297          0296  1 !       files are bypassed in the event of a file copy failure.
;  298          0297  1 !
;  299          0298  1 !--
;  300          0299  1
;  301          0300  1 !       NOTE:  This module contains some temporary code that (1) circumvents
;  302          0301  1 !       a system problem or (2) cannot be implemented until an expected
;  303          0302  1 !       system function is available.  In some cases, codes have been added;
;  304          0303  1 !       in other cases, code has been "commented out".  In either case, each
;  305          0304  1 !       statement affected includes a comment of the form "!#n", where "n" is
;  306          0305  1 !       a number from the following table:
;  307          0306  1 !
;  308          0307  1 !               #1 - symbol not currently defined in STARLET.L32
;  309          0308  1 !               #2 - I/O buffers cannot be locked in working set - known restriction
;  310          0309  1 !               #3 - MODIFY does not accept FHC XAB - future feature
```

```
;   312          0310  1 !
;   313          0311  1 ! TABLE OF CONTENTS:
;   314          0312  1 !
;   315          0313  1
;   316          0314  1 FORWARD ROUTINE
;   317          0315  1     COPY$COPY,                                  ! Main COPY control routine
;   318          0316  1     COPY$CHECK_FILE_FOR_MATCH,                  ! Sees if input file matches command line criteria
;   319          0317  1     CREATE_DIR,                                 ! Create directory file
;   320          0318  1     RMS_SETUP,                                  ! RAB/buffer initialization
;   321          0319  1     COPY_FILE,                                  ! Copies an input file to the output file
;   322          0320  1     CLOSE_INFILE : NOVALUE,                     ! Closes the current input file
;   323          0321  1     COPY$CLOSE_OUTF : NOVALUE,                  ! Closes the current output file
;   324          0322  1     BYPASS_CONCAT,                              ! Bypass concatenated input files after an error
;   325          0323  1     COPY$FIND_INPUT_FILE,                       ! Parse an input file-specification
;   326          0324  1     COPY$CALC_ALQ,                              ! Calculate the output file allocation quantity
;   327          0325  1     REPORT_NAMES : NOVALUE,                     ! Report names of input and output files
;   328          0326  1     REPORT_BYPASS : NOVALUE,                    ! Report name of file bypassed
;   329          0327  1     COPY$LOG_MSG : NOVALUE,                     ! Informational message routine
;   330          0328  1     COPY$INOPN_ERR : NOVALUE,                   ! Input open error routine
;   331          0329  1     IN_READ_ERROR : NOVALUE,                    ! Input read error routine
;   332          0330  1     IN_CLOSE_ERROR : NOVALUE,                   ! Input close error routine
;   333          0331  1     COPY$OUTOPN_ERR : NOVALUE,                  ! Output open error routine
;   334          0332  1     OUT_WRITE_ERROR : NOVALUE,                  ! Output write error routine
;   335          0333  1     COPY$OCLOSE_ERR : NOVALUE,                  ! Output close error routine
;   336          0334  1     COPY$MSG_NUMBER;                            ! Compute message number
;   337          0335  1
;   338          0336  1 !
;   339          0337  1 ! INCLUDE FILES:
;   340          0338  1 !
;   341          0339  1
;   342          0340  1 LIBRARY 'SYS$LIBRARY:STARLET.L32';              ! VAX/VMS common definitions
;   343          0341  1 REQUIRE 'SRC$:COPYMSG.REQ';                     ! Definition of macros to SIGNAL a message
;   344          0422  1
;   345          0423  1 !
;   346          0424  1 ! MACROS:
;   347          0425  1 !
;   348          0426  1
;   349          0427  1 MACRO
;   350        M 0428  1     IN_NEQ_OUT[] =                              ! Compare input and output FHC XAB field
;   351          0429  1         .INFILE_XABFHC[%REMAINING] NEQ .OUTFILE_XABFHC[%REMAINING] %,
;   352          0430  1
;   353          0431  1     NAM$B_DVILNG = $DEFINE_BYTE[NAM$T_DVI] %,
;   354          0432  1
;   355          0433  1     $DEFINE_BYTE( D, B, S, X ) = D, B, 8, 0 %,
;   356          0434  1
;   357          0435  1     !
;   358          0436  1     ! Check to see if the global or local qualifier flag is set without the
;   359          0437  1     ! local negation flag being set.
;   360          0438  1     !
;   361        M 0439  1     qualifier_active( global_qual, local_qual, locally_negated ) =
;   362        M 0440  1         (IF (.global_qual AND NOT .locally_negated) OR .local_qual
;   363        M 0441  1          THEN true
;   364          0442  1          ELSE false )%
;   365          0443  1     ;
;   366          0444  1
;   367          0445  1
;   368          0446  1 !
```

```
;  369          0447  1 ! EQUATED SYMBOLS:
;  370          0448  1 !
;  371          0449  1
;  372          0450  1 LITERAL
;  373          0451  1     CLI_STATUS_LEN = 28,                        ! Length of COPY$CLI_STATUS block
;  374          0452  1     SEM_STATUS_LEN =  4                         ! Length of COPY$SEM_STATUS block
;  375          0453  1     ;
;  376          0454  1 !    RME$K_OVERLAY = 0;                         !#1 ***** KLUDGE *****
;  377          0455  1
;  378          0456  1 !
;  379          0457  1 ! Global variables
;  380          0458  1 !
;  381          0459  1
;  382          0460  1 GLOBAL
;  383          0461  1     OUTFILE_COUNT :  INITIAL (0),               ! Number of output files created
;  384          0462  1
;  385          0463  1     BLOCK_COUNT,                                ! Number of input blocks copied (current file)
;  386          0464  1
;  387          0465  1     RECORD_COUNT,                               ! Number of input records copied (current file)
;  388          0466  1
;  389          0467  1     MOST_SEVERE_ERR :  BLOCK[4,BYTE]            ! Most severe error encountered
;  390          0468  1                        INITIAL( SS$_NORMAL ),   !
;  391          0469  1
;  392          0470  1     IO_BUFFER_BASE : INITIAL(0),                ! Address of I/O buffer pool
;  393          0471  1
;  394          0472  1     RMS_MBC : INITIAL(32),                      ! Size of the RMS buffers
;  395          0473  1
;  396          0474  1     BLOCK_SIZE,                                 ! Input file block size
;  397          0475  1
;  398          0476  1     COPY$CLI_STATUS : $BBLOCK[ CLI_STATUS_LEN ] ! Results of the command line parse
;  399          0477  1                       INITIAL(0),
;  400          0478  1     COPY$SEM_STATUS : $BBLOCK[ SEM_STATUS_LEN ] ! Status of the input and output files
;  401          0479  1                       INITIAL(0),
;  402          0480  1     COPY$B_INCOMPAT : BYTE INITIAL(0)           ! Flag which is set if files have incoompatible attr
;  403          0481  1     ;
;  404          0482  1
;  405          0483  1 !
;  406          0484  1 ! YET ANOTHER REQUIRE FILE
;  407          0485  1 !
;  408          0486  1 REQUIRE
;  409          0487  1     'SRC$:COPY.REQ';                            ! Field definitions for COPY$CLI_STATUS and COPY$SEM
```

```
; %PRINT:       File: VMSMAC.B32, Version V04-000, Edit 1, WWC, 09-JAN-1978
```

```
;  411          0942  1 !
;  412          0943  1 ! EXTERNAL REFERENCES:
;  413          0944  1 !
;  414          0945  1 EXTERNAL
;  415          0946  1
;  416          0947  1     !
;  417          0948  1     ! Command line qualifier values
;  418          0949  1     !
;  419          0950  1     common_qual_context,                        ! Common qualifier data area
;  420          0951  1     curr_allocation_value,                      ! The allocation for the output file
;  421          0952  1     curr_protection_or,                         ! Protection mask for /PROTECTION qualifier
;  422          0953  1     curr_protection_and,                        ! Protection mask for /PROTECTION qualifier
;  423          0954  1
;  424          0955  1     !
;  425          0956  1     ! RMS definitions
;  426          0957  1     !
;  427          0958  1     infile_fab          : BLOCK [, BYTE],       ! Input file FAB block
;  428          0959  1     infile_rab          : BLOCK [, BYTE],       ! Input file RAB block
;  429          0960  1     infile_name         : VECTOR [, BYTE],      ! Input file name after $OPEN
;  430          0961  1     infile_xname        : VECTOR [, BYTE],      ! Input file name before $OPEN
;  431          0962  1     infile_nam_blk      : BLOCK [, BYTE],       ! Primary input NAM block
;  432          0963  1     infile_xabfhc       : BLOCK [, BYTE],       ! File header characteristics XAB block
;  433          0964  1     infile_xaball       : BLOCK [, BYTE],       ! File allocation XAB block
;  434          0965  1     infile_cli_desc     : $BBLOCK,              ! Input file name on command line
;  435          0966  1     in_name_desc        : VECTOR,               ! Input file name descriptor
;  436          0967  1     outfile_fab         : BLOCK [, BYTE],       ! Output file FAB block
;  437          0968  1     outfile_rab         : BLOCK [, BYTE],       ! Output file RAB block
;  438          0969  1     outfile_name        : VECTOR [, BYTE],      ! Output file name after $OPEN
;  439          0970  1     outfile_xname       : VECTOR [, BYTE],      ! Output file name before $OPEN
;  440          0971  1     outfile_nam_blk     : BLOCK [, BYTE],       ! Output file NAM block
;  441          0972  1     outfile_xabrdt      : BLOCK [, BYTE],       ! Output file revision date/time XAB block
;  442          0973  1     outfile_xabpro      : BLOCK [, BYTE],       ! Output file protection XAB block
;  443          0974  1     outfile_xabdat      : BLOCK [, BYTE],       ! Output file date XAB block
;  444          0975  1     outfile_xaball      : BLOCK [, BYTE],       ! Output file allocation XAB block
;  445          0976  1     outfile_xabfhc      : BLOCK [, BYTE],       ! Output file file header characteristics XAB block
;  446          0977  1     out_name_desc       : VECTOR;               ! Output file name descriptor
;  447          0978  1
;  448          0979  1 EXTERNAL LITERAL
;  449          0980  1     LIB$_FILFAIMAT,                             ! File failed to match command line criteria
;  450          0981  1     LIB$_QUIPRO                                 ! User requested that processing cease
;  451          0982  1     ;
;  452          0983  1
;  453          0984  1 EXTERNAL ROUTINE
;  454          0985  1     COPY$GET_INFILE,                            ! Gets the name of the input file
;  455          0986  1     COPY$GET_OUTFIL,                            ! Gets the name of the output file
;  456          0987  1     COPY$OPN_INFILE,                            ! Opens the input file
;  457          0988  1     COPY$OPN_OUTFIL,                            ! Opens an output file
;  458          0989  1     CLI$GET_VALUE : ADDRESSING_MODE(GENERAL),   ! Get a value from the command line
;  459          0990  1     LIB$FIND_FILE : ADDRESSING_MODE(GENERAL),   ! Find a file which fits the given filespec
;  460          0991  1     LIB$GET_VM :  ADDRESSING_MODE(GENERAL),     ! Virtual memory allocation
;  461          0992  1     LIB$QUAL_FILE_MATCH : ADDRESSING_MODE(GENERAL),  ! Match a given file to the command line criteria
;  462          0993  1     LIB$CHECK_DIR :  ADDRESSING_MODE(GENERAL),  ! Determine if file is a directory
;  463          0994  1     LIB$CREATE_DIR :  ADDRESSING_MODE(GENERAL); ! Create a directory file
```

```
;  465     0995  1 ROUTINE COPY$COPY =                                        ! Primary COPY control routine
;  466     0996  1
;  467     0997  1 !++
;  468     0998  1 ! FUNCTIONAL DESCRIPTION:
;  469     0999  1 !
;  470     1000  1 !       This routine is the primary control routine for the COPY utility.
;  471     1001  1 !       It determines the basic logical flow and calls support routines
;  472     1002  1 !       which perform each logical function.
;  473     1003  1 !
;  474     1004  1 ! FORMAL PARAMETERS:
;  475     1005  1 !
;  476     1006  1 !       AP.rlu.va - Argument list passed from the Command Language Interpreter
;  477     1007  1 !
;  478     1008  1 ! IMPLICIT INPUTS:
;  479     1009  1 !
;  480     1010  1 !       None
;  481     1011  1 !
;  482     1012  1 ! IMPLICIT OUTPUTS:
;  483     1013  1 !
;  484     1014  1 !       None
;  485     1015  1 !
;  486     1016  1 ! COMPLETION CODES:
;  487     1017  1 !
;  488     1018  1 !       Most severe error encountered during processing or SS$_NORMAL
;  489     1019  1 !
;  490     1020  1 ! SIDE EFFECTS:
;  491     1021  1 !
;  492     1022  1 !       None
;  493     1023  1 !
;  494     1024  1 !--
;  495     1025  1
;  496     1026  2     BEGIN
;  497     1027  2
;  498     1028  2     BUILTIN
;  499     1029  2         AP;                                                 ! Declare the name of the argument pointer.
;  500     1030  2
;  501     1031  2     BIND
;  502     1032  2         ARGUMENT_LIST = AP :  REF BLOCK[,BYTE];             ! Declare the form of the argument list.
;  503     1033  2
;  504     1034  2     LOCAL
;  505     1035  2         ptr,                                                ! Temporary variables for character searching
;  506     1036  2         address,
;  507     1037  2         size,
;  508     1038  2         STATUS;                                             ! General routine return code
;  509     1039  2
;  510     1040  2
;  511     1041  2
;  512     1042  2
;  513     1043  2
;  514     1044  2 !
;  515     1045  2 ! Get the output file-specification and all qualifiers from the CLI.
;  516     1046  2 !
;  517     1047  2
;  518     1048  2     IF NOT COPY$GET_OUTFIL (                                ! Get the output file spec from the CLI.
;  519     1049  2                             OUTFILE_FAB,                   !    Specify the output FAB block address,
;  520     1050  2                             OUTFILE_NAM_BLK,               !    the output NAM block address,
;  521     1051  2                             OUTFILE_XABFHC)                !    and the output XABFHC block address.
```

```
;  522          1052  2      THEN
;  523          1053  2          RETURN .MOST_SEVERE_ERR;                        !    On error, return to CLI.
;  524          1054  2
;  525          1055  2  !
;  526          1056  2  ! The remainder of this routine is executed for each input
;  527          1057  2  ! file-specification supplied by the user.  Get the first input file.
;  528          1058  2  !
;  529          1059  2
;  530          1060  2      IF NOT (status = CLI$GET_VALUE( $DESCRIPTOR('INFILE'), infile_cli_desc))
;  531          1061  2      THEN
;  532          1062  2          RETURN .status;
;  533          1063  2
;  534          1064  2      WHILE 1 DO                                          ! Beginning of repeat loop
;  535          1065  2          BEGIN
;  536          1066  2
;  537          1067  2  !
;  538          1068  2  ! Get the next input file-specification from the CLI.  This routine call is a
;  539          1069  2  ! NOP if a wildcard file-specification is currently being processed;
;  540          1070  2  ! that is, a wildcard specification is repeatedly used until no furthur
;  541          1071  2  ! match is found.
;  542          1072  2  !
;  543          1073  2
;  544          1074  2          STATUS = COPY$GET_INFILE (                      ! Get an input file-specification.
;  545          1075  2                                    INFILE_FAB,           !    Specify the address of the input FAB block,
;  546          1076  2                                    INFILE_NAM_BLK,       !    the address of the input NAM block,
;  547          1077  2                                    INFILE_XABALL);       !    and the address of the input XABALL block.
;  548          1078  2
;  549          1079  2          IF .STATUS EQL NO_MORE_FILES                    ! If there are no more input file-specs,
;  550          1080  2          THEN                                            !
;  551          1081  2              EXITLOOP;                                   ! exit the input file-spec processing loop.
;  552          1082  2
;  553          1083  2          IF .STATUS EQL OK                               ! If everything is OK so far,
;  554          1084  2          THEN                                            !
;  555          1085  3              BEGIN                                       ! begin normal input file processing.
;  556          1086  3
;  557          1087  3  !
;  558          1088  3  ! Open the current input file.
;  559          1089  3  !
;  560          1090  3
;  561          1091  3              STATUS = COPY$OPN_INFILE (INFILE_FAB);      ! Open the current input file.
;  562          1092  3
;  563          1093  3  !
;  564          1094  3  ! If the input file is a directory file, then create the directory file
;  565          1095  3  ! on the output side if the file does not already exist.  If the output
;  566          1096  3  ! directory already exists, then do nothing.
;  567          1097  3  !
;  568          1098  3
;  569          1099  3              IF .status EQL ok                           ! If input opened ok,
;  570          1100  3                  AND lib$check_dir (infile_fab)          ! and file is a directory,
;  571          1101  3                  AND NOT .outfile_fab [$FAB_DEV(sdi)]    ! and not magtape output,
;  572          1102  3              THEN
;  573          1103  3                  IF NOT .outfile_fab[$FAB_DEV(net)]
;  574          1104  3                      AND NOT .outfile_fab [$FAB_DEV(rec)] ! and not record device,
;  575          1105  3                  THEN
;  576          1106  4                          BEGIN
;  577          1107  5                          IF (.outfile_nam_blk[nam$v_exp_name] AND
;  578          1108  4                                  (NOT .outfile_nam_blk[nam$v_wild_name])) OR
```

```
;  579          1109  6                    (.outfile_nam_blk[nam$v_exp_type] AND
;  580          1110  5                        (NOT .outfile_nam_blk[nam$v_wild_type])) OR
;  581          1111  6                    (.outfile_nam_blk[nam$v_exp_ver] AND
;  582          1112  6                        (NOT .outfile_nam_blk[nam$v_wild_ver]))
;  583          1113  5                THEN
;  584          1114  6                    BEGIN
;  585          1115  6                    report_bypass(msg$_illdircopy);
;  586          1116  6                    close_infile();                 ! Close input file
;  587          1117  6                    END
;  588          1118  5                ELSE
;  589          1119  6                    BEGIN
;  590          1120  6                    status = create_dir (infile_fab, outfile_fab);
;  591          1121  6                    IF .status EQL ss$_created  ! If file actually created,
;  592          1122  9                    THEN
;  593          1123  7                        BEGIN
;  594          1124  7                        report_names();         ! Report file copied
;  595          1125  7                        outfile_count = .outfile_count + 1;
;  596          1126  6                        END;
;  597          1127  6                    IF NOT .status                  ! If successful,
;  598          1128  6                    THEN
;  599          1129  6                        report_bypass(msg$_notcopied); ! Else report failure
;  600          1130  6                    close_infile();                 ! Close input file
;  601          1131  6                    END
;  602          1132  5                END
;  603          1133  4            ELSE
;  604          1134  5                BEGIN
;  605          1135  5                report_bypass(msg$_dirnotcre);  ! Else report failure
;  606          1136  5                close_infile();                 ! Close input file
;  607          1137  5                END
;  608          1138  4            ELSE
;  609          1139  5            BEGIN
;  610          1140  5
;  611          1141  5
;  612          1142  5        !
;  613          1143  5        ! Create (or simply open) the output file (if it is not already open due to
;  614          1144  5        ! input file concatenation) and then copy the entire input file to the
;  615          1145  5        ! output file.
;  616          1146  5        !
;  617          1147  5
;  618          1148  5                IF .STATUS EQL OK                       ! If the input file was successfully opened,
;  619          1149  5                THEN                                    !
;  620          1150  6                    BEGIN                               !
;  621          1151  7                    IF (STATUS = COPY$OPN_OUTFIL (      ! create or open the output file unless it is
;  622          1152  7                                                  OUTFILE_FAB,
;  623          1153  7                                                  OUTFILE_RAB,
;  624          1154  7                                                  INFILE_FAB,
;  625          1155  7                                                  OUTFILE_COUNT))
;  626          1156  7                                                  ! already open due to input concatenation.
;  627          1157  6                        THEN
;  628          1158  7                            BEGIN
;  629          1159  8                            IF (STATUS = RMS_SETUP())       ! Setup the input and output RABs and buffers.
;  630          1160  7                            THEN
;  631          1161  8                                BEGIN
;  632          1162  9                                IF (STATUS = COPY_FILE())   ! Copy the entire input file to the output file.
;  633          1163  8                                THEN
;  634          1164  9                                    BEGIN
;  635          1165  9                                    IF .outfile_fab [$FAB_DEV(rec)]
```

```
;  636          1166  9                        AND NOT .outfile_fab [$FAB_DEV(net)]
;  637          1167  9                    THEN
;  638          1168 10                        BEGIN
;  639          1169 10                        size = .out_name_desc[0];
;  640          1170 10                        address = .out_name_desc[1];
;  641          1171 10                        ptr = CH$FIND_CH(.size,.address,':');
;  642          1172 10                        IF .ptr NEQ 0           ! If there is anything past the device, remove it
;  643          1173 10                        THEN
;  644          1174 10                            out_name_desc[0] =  .ptr - .address + 1;
;  645          1175  9                        END;
;  646          1176  9                    REPORT_NAMES()              ! Report the results if the copy was successful.
;  647          1177  9                    END
;  648          1178  8                ELSE                            ! Otherwise, report a partial copy.
;  649          1179  8                    REPORT_BYPASS( MSG$_NOTCMPLT );
;  650          1180  8                END
;  651          1181  7            ELSE
;  652          1182  7                REPORT_BYPASS( MSG$_NOTCOPIED );
;  653          1183  7            END
;  654          1184  6        ELSE                                    ! If the output file couldn't be opened,
;  655          1185  7            BEGIN
;  656          1186  7
;  657          1187  7            ! If this is an APPEND operation, then stop processing.
;  658          1188  7            ! There is no need to continue appending to a non-existant
;  659          1189  7            ! file.
;  660          1190  7            !
;  661          1191  7            IF .append_command
;  662          1192  7            THEN EXITLOOP;
;  663          1193  7
;  664          1194  7            SELECTONE .status OF
;  665          1195  7            SET
;  666          1196  7                [ LIB$_FILFAIMAT ] :    ! Quietly skip this file
;  667          1197  7                    status = ok;
;  668          1198  7                [ LIB$_QUIPRO ]    :    ! User wishes to stop at this point
;  669          1199  7                    EXITLOOP;
;  670          1200  7                [ OTHERWISE ]      :    ! indicate the input file wasn't copied.
;  671          1201  7                    REPORT_BYPASS( MSG$_NOTCOPIED);
;  672          1202  7            TES;
;  673          1203  6            END; ! else stmt
;  674          1204  5        END;
;  675          1205  5
;  676          1206  5        CLOSE_INFILE();                         ! Close the input file.
;  677          1207  5
;  678          1208  4        END;                                    ! End of ELSE clause
;  679          1209  3        END;                                    ! End of processing a single input file specificatio
;  680          1210  3
;  681          1211  3
;  682          1212  3    ! If the user wishes to quit processing, then exit with a successful
;  683          1213  3    ! status.
;  684          1214  3    !
;  685          1215  3    IF .status EQL LIB$_QUIPRO
;  686          1216  3    THEN
;  687          1217  3        status = ok;
;  688          1218  3    !
;  689          1219  3    ! Bypass any concatenated input files if an error occurred during the
;  690          1220  3    ! file copy.
;  691          1221  3    !
;  692          1222  3
```

```
;  693          1223  3          IF NOT .STATUS                                  ! If the input file was not successfully copied,
;  694          1224  3          THEN                                            !
;  695          1225  3              IF BYPASS_CONCAT()                          ! bypass any concatenated input files.
;  696          1226  3              THEN
;  697          1227  3                  EXITLOOP;
;  698          1228  3
;  699          1229  3      !
;  700          1230  3      ! Close the output file unless another input file is to be
;  701          1231  3      ! concatenated to the output file just written.
;  702          1232  3      !
;  703          1233  3
;  704          1234  3          IF .MULTIPLE_OUTPUT AND NOT .APPEND_COMMAND     ! If multiple output files are being created,
;  705          1235  4          THEN                                            !    and the command was not APPEND,
;  706          1236  4              BEGIN
;  707          1237  4              !
;  708          1238  4              ! Set up protection if user specified explicitly.
;  709          1239  4              !
;  710          1240  5              IF qualifier_active( protect_qual, loc_protect_qual, neg_protect_qual )
;  711          1241  4              THEN
;  712          1242  5                  BEGIN
;  713          1243  5                  outfile_xabpro [xab$w_pro] = .outfile_xabpro[ xab$w_pro] AND
;  714          1244  5                                               .curr_protection_and;
;  715          1245  5                  outfile_xabpro [xab$w_pro] = .outfile_xabpro[ xab$w_pro] OR
;  716          1246  5                                               .curr_protection_or;
;  717          1247  5                  END
;  718          1248  4              ELSE
;  719          1249  4                  outfile_xabrdt [xab$l_nxt] = 0;
;  720          1250  4
;  721          1251  4              !
;  722          1252  4              ! Close the current output file.
;  723          1253  4              !
;  724          1254  4
;  725          1255  4              COPY$CLOSE_OUTF();                          ! close the current output file, if any.
;  726          1256  4
;  727          1257  4              !
;  728          1258  4              ! Reinitialize the XAB chain, since it may have been mucked with
;  729          1259  4              ! by COPY$OPN_OUTFIL among other routines.
;  730          1260  4              !
;  731          1261  4
;  732          1262  4              outfile_xaball [xab$l_nxt] = outfile_xabdat;
;  733          1263  4              outfile_xabdat [xab$l_nxt] = outfile_xabrdt;
;  734          1264  4              outfile_xabrdt [xab$l_nxt] = outfile_xabpro;
;  735          1265  3              END;
;  736          1266  3
;  737          1267  2          END;                                            ! End of 'WHILE 1 DO' input file-spec processing loo
;  738          1268  2
;  739          1269  2      !
;  740          1270  2      ! Perform any necessary cleanup before exiting.
;  741          1271  2      !
;  742          1272  2
;  743          1273  2          !
;  744          1274  2          ! Set up protection if user specified explicitly.
;  745          1275  2          !
;  746          1276  2
;  747          1277  3          IF qualifier_active( protect_qual, loc_protect_qual, neg_protect_qual )
;  748          1278  2          THEN
;  749          1279  3              BEGIN
```

```
;  750          1280  3          outfile_xabpro [xab$w_pro] = .outfile_xabpro[ xab$w_pro] AND
;  751          1281  3                                       .curr_protection_and;
;  752          1282  3          outfile_xabpro [xab$w_pro] = .outfile_xabpro[ xab$w_pro] OR
;  753          1283  3                                       .curr_protection_or;
;  754          1284  3          END
;  755          1285  2      ELSE
;  756          1286  2          outfile_xabrdt [xab$l_nxt] = 0;
;  757          1287  2
;  758          1288  2      COPY$CLOSE_OUTF();                          ! close the current output file, if any.
;  759          1289  2
;  760          1290  2      COPY$LOG_MSG( MSG$_NEWFILES );              ! Report the number of files created.
;  761          1291  2
;  762          1292  2  !
;  763          1293  2  ! Return to the caller.
;  764          1294  2  !
;  765          1295  2
;  766          1296  2      RETURN .MOST_SEVERE_ERR;                    ! Use the most severe error encountered
;  767          1297  2                                                  ! as the completion code from this routine.
;  768          1298  2
;  769          1299  1      END;


                                                          .TITLE  COPYMAIN
                                                          .IDENT  \V04-000\

                                                          .PSECT  $PLIT$,NOWRT,NOEXE,2

                                45  4C  49  46  4E  49   00000 P.AAB:   .ASCII  \INFILE\                                 ;
                                                    00006               .BLKB   2
                                          00000006  00008 P.AAA:        .LONG   6                                       ;
                                          00000000' 0000C               .ADDRESS P.AAB                                  ;

                                                          .PSECT  $GLOBAL$,NOEXE,2

                                00000000  00000 OUTFILE_COUNT::
                                                          .LONG   0                                                     ;
                                          00004 BLOCK_COUNT::
                                                          .BLKB   4
                                          00008 RECORD_COUNT::
                                                          .BLKB   4
                                00000001  0000C MOST_SEVERE_ERR::
                                                          .LONG   1                                                     ;
                                00000000  00010 IO_BUFFER_BASE::
                                                          .LONG   0                                                     ;
                                00000020  00014 RMS_MBC::
                                                          .LONG   32                                                    ;
                                          00018 BLOCK_SIZE::
                                                          .BLKB   4
                                00000000  0001C COPY$CLI_STATUS::
                                                          .LONG   0                                                     ;
                                          00020           .BLKB   24
                                00000000  00038 COPY$SEM_STATUS::
                                                          .LONG   0                                                     ;
                                      00  0003C COPY$B_INCOMPAT::
                                                          .BYTE   0                                                     ;

                                                          .EXTRN  CLI$_PRESENT, CLI$_NEGATED
```

```
                                .EXTRN  CLI$_LOCPRES, CLI$_LOCNEG
                                .EXTRN  COMMON_QUAL_CONTEXT
                                .EXTRN  CURR_ALLOCATION_VALUE
                                .EXTRN  CURR_PROTECTION_OR
                                .EXTRN  CURR_PROTECTION_AND
                                .EXTRN  INFILE_FAB, INFILE_RAB
                                .EXTRN  INFILE_NAME, INFILE_XNAME
                                .EXTRN  INFILE_NAM_BLK, INFILE_XABFHC
                                .EXTRN  INFILE_XABALL, INFILE_CLI_DESC
                                .EXTRN  IN_NAME_DESC, OUTFILE_FAB
                                .EXTRN  OUTFILE_RAB, OUTFILE_NAME
                                .EXTRN  OUTFILE_XNAME, OUTFILE_NAM_BLK
                                .EXTRN  OUTFILE_XABRDT, OUTFILE_XABPRO
                                .EXTRN  OUTFILE_XABDAT, OUTFILE_XABALL
                                .EXTRN  OUTFILE_XABFHC, OUT_NAME_DESC
                                .EXTRN  LIB$_FILFAIMAT, LIB$_QUIPRO
                                .EXTRN  COPY$GET_INFILE
                                .EXTRN  COPY$GET_OUTFIL
                                .EXTRN  COPY$OPN_INFILE
                                .EXTRN  COPY$OPN_OUTFIL
                                .EXTRN  CLI$GET_VALUE, LIB$FIND_FILE
                                .EXTRN  LIB$GET_VM, LIB$QUAL_FILE_MATCH
                                .EXTRN  LIB$CHECK_DIR, LIB$CREATE_DIR

                                .PSECT  $CODE$,NOWRT,2

                        0FFC 00000 COPY$COPY:
                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11            ; 0995
5B      0000G   CF  9E  00002           MOVAB   OUTFILE_XABPRO+8, R11                   ;
5A      0000G   CF  9E  00007           MOVAB   INFILE_FAB, R10                         ;
59      0000G   CF  9E  0000C           MOVAB   OUTFILE_NAM_BLK+52, R9                  ;
58      0000G   CF  9E  00011           MOVAB   OUTFILE_FAB+64, R8                      ;
57      0000'   CF  9E  00016           MOVAB   COPY$CLI_STATUS+2, R7                   ;
        0000G   CF  9F  0001B           PUSHAB  OUTFILE_XABFHC                          ; 1048
          CC    A9  9F  0001F           PUSHAB  OUTFILE_NAM_BLK                         ;
          C0    AB  9F  00022           PUSHAB  OUTFILE_FAB                             ;
0000G   CF              03  FB  00025   CALLS   #3, COPY$GET_OUTFIL                     ;
          03            50  E8  0002A   BLBS    R0, 1$                                ;
                      01D1  31  0002D   BRW     38$                                   ;
        0000G   CF  9F  00030 1$:       PUSHAB  INFILE_CLI_DESC                         ; 1060
        0000'   CF  9F  00034           PUSHAB  P.AAA                                   ;
00000000G   00          02  FB  00038   CALLS   #2, CLI$GET_VALUE                      ;
          52            50  D0  0003F   MOVL    R0, STATUS                            ;
          04            52  E8  00042   BLBS    STATUS, 2$                            ;
          50            52  D0  00045   MOVL    STATUS, R0                            ; 1062
                        04  00048       RET                                           ;
        0000G   CF  9F  00049 2$:       PUSHAB  INFILE_XABALL                           ; 1074
        0000G   CF  9F  0004D           PUSHAB  INFILE_NAM_BLK                          ;
                5A  DD  00051           PUSHL   R10                                     ;
0000G   CF              03  FB  00053   CALLS   #3, COPY$GET_INFILE                    ;
          52            50  D0  00058   MOVL    R0, STATUS                            ;
          03            52  D1  0005B   CMPL    STATUS, #3                            ; 1079
                        03  12  0005E   BNEQ    3$                                    ;
                      016D  31  00060   BRW     33$                                   ;
          01            52  D1  00063 3$:       CMPL    STATUS, #1                    ; 1083
                        03  13  00066   BEQL    4$                                    ;
                      0108  31  00068   BRW     25$                                   ;
```

```
                                        5A   DD 0006B 4$:     PUSHL   R10                                     ; 1091
                    0000G   CF          01   FB 0006D         CALLS   #1, COPY$OPN_INFILE                     ;
                            52          50   D0 00072         MOVL    R0, STATUS                              ;
                                        53   D4 00075         CLRL    R3                                      ; 1099
                            01          52   D1 00077         CMPL    STATUS, #1                              ;
                                        63   12 0007A         BNEQ    12$                                     ;
                                        53   D6 0007C         INCL    R3                                      ;
                                        5A   DD 0007E         PUSHL   R10                                     ; 1100
                00000000G   00          01   FB 00080         CALLS   #1, LIB$CHECK_DIR                       ;
                            55          50   E9 00087         BLBC    R0, 12$                                 ;
                51          68          04   E0 0008A         BBS     #4, OUTFILE_FAB+64, 12$                 ; 1101
                45      01  A8          05   E0 0008E         BBS     #5, OUTFILE_FAB+65, 10$                 ; 1103
                            42          68   E8 00093         BLBS    OUTFILE_FAB+64, 10$                     ; 1104
                04          69          02   E1 00096         BBC     #2, OUTFILE_NAM_BLK+52, 5$              ; 1107
                0F          69          05   E1 0009A         BBC     #5, OUTFILE_NAM_BLK+52, 7$              ; 1108
                04          69          01   E1 0009E 5$:     BBC     #1, OUTFILE_NAM_BLK+52, 6$              ; 1109
                07          69          04   E1 000A2         BBC     #4, OUTFILE_NAM_BLK+52, 7$              ; 1110
                            0B          69   E9 000A6 6$:     BLBC    OUTFILE_NAM_BLK+52, 8$                  ; 1111
                07          69          03   E0 000A9         BBS     #3, OUTFILE_NAM_BLK+52, 8$              ; 1112
                            7E  12E8    8F   3C 000AD 7$:     MOVZWL  #4840, -(SP)                            ; 1115
                                        29   11 000B2         BRB     11$                                     ;
                                C0      A8   9F 000B4 8$:     PUSHAB  OUTFILE_FAB                             ; 1120
                                        5A   DD 000B7         PUSHL   R10                                     ;
                    0000V   CF          02   FB 000B9         CALLS   #2, CREATE_DIR                          ;
                            52          50   D0 000BE         MOVL    R0, STATUS                              ;
                00000619    8F          52   D1 000C1         CMPL    STATUS, #1561                           ; 1121
                                        08   12 000C8         BNEQ    9$                                      ;
                    0000V   CF          00   FB 000CA         CALLS   #0, REPORT_NAMES                        ; 1124
                                E2      A7   D6 000CF         INCL    OUTFILE_COUNT                           ; 1125
                            68          52   E8 000D2 9$:     BLBS    STATUS, 15$                             ; 1127
                                    008C    31   000D5         BRW     22$                                     ; 1129
                            7E  12C0    8F   3C 000D8 10$:    MOVZWL  #4800, -(SP)                            ; 1135
                                        65   11 000DD 11$:    BRB     17$                                     ;
                            77          53   E9 000DF 12$:    BLBC    R3, 20$                                 ; 1148
                                E2      A7   9F 000E2         PUSHAB  OUTFILE_COUNT                           ; 1151
                                        5A   DD 000E5         PUSHL   R10                                     ;
                                0000G   CF   9F 000E7         PUSHAB  OUTFILE_RAB                             ;
                                C0      A8   9F 000EB         PUSHAB  OUTFILE_FAB                             ;
                    0000G   CF          04   FB 000EE         CALLS   #4, COPY$OPN_OUTFIL                     ;
                            52          50   D0 000F3         MOVL    R0, STATUS                              ;
                            4D          52   E9 000F6         BLBC    STATUS, 18$                             ;
                    0000V   CF          00   FB 000F9         CALLS   #0, RMS_SETUP                           ; 1159
                            52          50   D0 000FE         MOVL    R0, STATUS                              ;
                            60          52   E9 00101         BLBC    STATUS, 22$                             ;
                    0000V   CF          00   FB 00104         CALLS   #0, COPY_FILE                           ; 1162
                            52          50   D0 00109         MOVL    R0, STATUS                              ;
                            30          52   E9 0010C         BLBC    STATUS, 16$                             ;
                            26          68   E9 0010F         BLBC    OUTFILE_FAB+64, 14$                     ; 1165
                21      01  A8          05   E0 00112         BBS     #5, OUTFILE_FAB+65, 14$                 ; 1166
                            55  0000G   CF   D0 00117         MOVL    OUT_NAME_DESC, SIZE                     ; 1169
                            54  0000G   CF   D0 0011C         MOVL    OUT_NAME_DESC+4, ADDRESS                ; 1170
                64          55          3A   3A 00121         LOCC    #58, SIZE, (ADDRESS)                    ; 1171
                                        02   12 00125         BNEQ    13$                                     ;
                                        51   D4 00127         CLRL    R1                                      ;
                            56          51   D0 00129 13$:    MOVL    R1, PTR                                 ;
                                        0A   13 0012C         BEQL    14$                                     ; 1172
                51          56          54   C3 0012E         SUBL3   ADDRESS, PTR, R1                        ; 1174
```

```
                0000G  CF        01    A1  9E 00132             MOVAB   1(R1), OUT_NAME_DESC                          ;
                0000V  CF              00  FB 00138 14$:        CALLS   #0, REPORT_NAMES                            ; 1176
                                       2F  11 0013D 15$:        BRB     24$                                         ;
                       7E      11C0    8F  3C 0013F 16$:        MOVZWL  #4544, -(SP)                                ; 1179
                                       23  11 00144 17$:        BRB     23$                                         ;
                       03        FE    A7  E9 00146 18$:        BLBC    COPY$CLI_STATUS, 19$                        ; 1191
                                 0083  31  00149                BRW     33$                                         ;
        00000000G      8F              52  D1 0014D 19$:        CMPL    STATUS, #LIB$_FILFAIMAT                     ; 1196
                                       05  12 00154             BNEQ    21$                                         ;
                       52              01  D0 00156             MOVL    #1, STATUS                                  ; 1197
                                       13  11 00159 20$:        BRB     24$                                         ;
        00000000G      8F              52  D1 0015B 21$:        CMPL    STATUS, #LIB$_QUIPRO                        ; 1198
                                       6C  13 00162             BEQL    33$                                         ;
                       7E      11B8    8F  3C 00164 22$:        MOVZWL  #4536, -(SP)                                ; 1201
                0000V  CF              01  FB 00169 23$:        CALLS   #1, REPORT_BYPASS                           ;
                0000V  CF              00  FB 0016E 24$:        CALLS   #0, CLOSE_INFILE                            ; 1206
        00000000G      8F              52  D1 00173 25$:        CMPL    STATUS, #LIB$_QUIPRO                        ; 1215
                                       03  12 0017A             BNEQ    26$                                         ;
                       52              01  D0 0017C             MOVL    #1, STATUS                                  ; 1217
                       08              52  E8 0017F 26$:        BLBS    STATUS, 27$                                 ; 1223
                0000V  CF              00  FB 00182             CALLS   #0, BYPASS_CONCAT                           ; 1225
                       46              50  E8 00187             BLBS    R0, 33$                                     ;
                       3F        1B    A7  E9 0018A 27$:        BLBC    COPY$SEM_STATUS+1, 32$                      ; 1233
                       3B        FE    A7  E8 0018E             BLBS    COPY$CLI_STATUS, 32$                        ;
          05      01   A7              05  E1 00192             BBC     #5, COPY$CLI_STATUS+3, 28$                  ; 1240
                                 01    A7  95 00197             TSTB    COPY$CLI_STATUS+3                           ;
                                       05  18 0019A             BGEQ    29$                                         ;
          0F      01   A7              06  E1 0019C 28$:        BBC     #6, COPY$CLI_STATUS+3, 30$                  ;
                       50      0000G  CF  D2 001A1 29$:        MCOML   CURR_PROTECTION_AND, R0                     ; 1244
                       6B              50  AA 001A6             BICW2   R0, OUTFILE_XABPRO+8                        ;
                       6B      0000G  CF  A8 001A9             BISW2   CURR_PROTECTION_OR, OUTFILE_XABPRO+8         ; 1246
                                       04  11 001AE             BRB     31$                                         ; 1240
                              0000G    CF  D4 001B0 30$:        CLRL    OUTFILE_XABRDT+4                            ; 1249
                       0000V  CF              00  FB 001B4 31$:        CALLS   #0, COPY$CLOSE_OUTF                 ; 1255
                       0000G  CF      0000G   CF  9E 001B9             MOVAB   OUTFILE_XABDAT, OUTFILE_XABALL+4    ; 1262
                       0000G  CF      0000G   CF  9E 001C0             MOVAB   OUTFILE_XABRDT, OUTFILE_XABDAT+4    ; 1263
                       0000G  CF        F8    AB  9E 001C7             MOVAB   OUTFILE_XABPRO, OUTFILE_XABRDT+4    ; 1264
                                     FE79    31  001CD 32$:        BRW     2$                                          ; 1064
          05      01   A7              05  E1 001D0 33$:        BBC     #5, COPY$CLI_STATUS+3, 34$                  ; 1277
                                 01    A7  95 001D5             TSTB    COPY$CLI_STATUS+3                           ;
                                       05  18 001D8             BGEQ    35$                                         ;
          0F      01   A7              06  E1 001DA 34$:        BBC     #6, COPY$CLI_STATUS+3, 36$                  ;
                       50      0000G  CF  D2 001DF 35$:        MCOML   CURR_PROTECTION_AND, R0                     ; 1281
                       6B              50  AA 001E4             BICW2   R0, OUTFILE_XABPRO+8                        ;
                       6B      0000G  CF  A8 001E7             BISW2   CURR_PROTECTION_OR, OUTFILE_XABPRO+8         ; 1283
                                       04  11 001EC             BRB     37$                                         ; 1277
                              0000G    CF  D4 001EE 36$:        CLRL    OUTFILE_XABRDT+4                            ; 1286
                       0000V  CF              00  FB 001F2 37$:        CALLS   #0, COPY$CLOSE_OUTF                 ; 1288
                       7E      1091    8F  3C 001F7             MOVZWL  #4241, -(SP)                                ; 1290
                0000V  CF              01  FB 001FC             CALLS   #1, COPY$LOG_MSG                            ;
                       50        EE    A7  D0 00201 38$:        MOVL    MOST_SEVERE_ERR, R0                         ; 1296
                                       04  00205             RET                                                 ; 1299

; Routine Size:  518 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  771          1300  1 GLOBAL ROUTINE COPY$CHECK_FILE_FOR_MATCH =
;  772          1301  1
;  773          1302  1 !++
;  774          1303  1 !
;  775          1304  1 ! FUNCTIONAL DESCRIPTION:
;  776          1305  1 !
;  777          1306  1 !       This routine sets up the parameters for and calls LIB$QUAL_FILE_MATCH to see if the input
;  778          1307  1 !       file matches the criteria given on the command line.
;  779          1308  1 !
;  780          1309  1 ! FORMAL PARAMETERS:
;  781          1310  1 !
;  782          1311  1 !       None
;  783          1312  1 !
;  784          1313  1 ! IMPLICIT INPUTS:
;  785          1314  1 !
;  786          1315  1 !       IN_NAME_DESC    : Input file name descriptor
;  787          1316  1 !       OUT_NAME_DESC   : Output file name descriptor
;  788          1317  1 !       OUTFILE_OPEN    : Output file is currently open
;  789          1318  1 !       COMMON_QUAL_CONTEXT : Common qualifier data area
;  790          1319  1 !
;  791          1320  1 ! IMPLICIT OUTPUTS:
;  792          1321  1 !
;  793          1322  1 !       None
;  794          1323  1 !
;  795          1324  1 ! ROUTINE VALUE:
;  796          1325  1 !
;  797          1326  1 !       Whatever LIB$QUAL_FILE_MATCH returns.
;  798          1327  1 !
;  799          1328  1 ! COMPLETION CODES:
;  800          1329  1 !
;  801          1330  1 !       None
;  802          1331  1 !
;  803          1332  1 ! SIDE EFFECTS:
;  804          1333  1 !
;  805          1334  1 !       None
;  806          1335  1 !
;  807          1336  1 !--
;  808          1337  1
;  809          1338  2 BEGIN
;  810          1339  2
;  811          1340  2 LOCAL
;  812          1341  2     out_desc      :                       ! Temporary desc. for output file name
;  813          1342  2                     VECTOR[ 2 ],
;  814          1343  2     prompt_string_desc,                   ! Desc. for /CONFIRM prompt string address
;  815          1344  2     prompt_args   :                       ! Argument list for /CONFIRM prompt
;  816          1345  2                     VECTOR[ 2 ]
;  817          1346  2     ;
;  818          1347  2
;  819          1348  2
;  820          1349  2
;  821          1350  2 ! Pick to appropriate propmt string, depending on whether the input file is
;  822          1351  2 ! being append to an output file or not.
;  823          1352  2 !
;  824          1353  2 IF .append_command OR  .outfile_open
;  825          1354  2     THEN prompt_string_desc = $DESCRIPTOR('Append !AS to !AS? [N]')
;  826          1355  2     ELSE prompt_string_desc = $DESCRIPTOR('Copy !AS to !AS? [N]');
;  827          1356  2
```

```
;  828          1357  2
;  829          1358  2 ! File in the file name descriptors.
;  830          1359  2 !
;  831          1360  2 prompt_args[ 0 ] = in_name_desc;
;  832          1361  2 prompt_args[ 1 ] = out_desc;
;  833          1362  2
;  834          1363  2 IF .outfile_nam_blk[ NAM$B_RSL ] NEQ 0
;  835          1364  2 THEN
;  836          1365  3     BEGIN
;  837          1366  3     out_desc[ 0 ] = .outfile_nam_blk[ NAM$B_RSL ];
;  838          1367  3     out_desc[ 1 ] = outfile_name;
;  839          1368  3     END
;  840          1369  2 ELSE
;  841          1370  3     IF .outfile_nam_blk[ NAM$B_ESL ] NEQ 0
;  842          1371  3     THEN
;  843          1372  4         BEGIN
;  844          1373  4         out_desc[ 0 ] = .outfile_nam_blk[ NAM$B_ESL ];
;  845          1374  4         out_desc[ 1 ] = outfile_xname;
;  846          1375  4         END
;  847          1376  3     ELSE
;  848          1377  3         prompt_args[ 1 ] = out_name_desc;
;  849          1378  3
;  850          1379  2
;  851          1380  2 ! Compare the current input file to the command line criteria.  Return the
;  852          1381  2 ! results of the comparison to the calling routine.
;  853          1382  2 !
;  854          1383  2 RETURN LIB$QUAL_FILE_MATCH( common_qual_context, infile_fab, 0,
;  855          1384  2     .prompt_string_desc, prompt_args, 0);
;  856          1385  2
;  857          1386  1 END;                                     ! End of routine COPY$CHECK_FILE_FOR_MATCH


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

21 20 6F 74 20 53 41 21 20 64 6E 65 70 70 41  00010 P.AAD:   .ASCII  \Append !AS to !AS? [N]\                    ;
                        5D 4E 5B 20 3F 53 41  0001F                                                             ;
                                              00026         .BLKB   2                                           ;
                                    00000016  00028 P.AAC:  .LONG   22                                          ;
                                    00000000' 0002C         .ADDRESS P.AAD                                      ;
53 41 21 20 6F 74 20 53 41 21 20 79 70 6F 43  00030 P.AAF:   .ASCII  \Copy !AS to !AS? [N]\                     ;
                                 5D 4E 5B 20 3F  0003F                                                          ;
                                    00000014  00044 P.AAE:  .LONG   20                                          ;
                                    00000000' 00048         .ADDRESS P.AAF                                      ;


                                                                .PSECT  $CODE$,NOWRT,2

                                         0000 00000             .ENTRY  COPY$CHECK_FILE_FOR_MATCH, Save nothing ; 1300
                                 5E        10  C2 00002         SUBL2   #16, SP                                 ;
                                 06  0000' CF  E8 00005         BLBS    COPY$CLI_STATUS, 1$                     ; 1353
                           07    0000' CF     01  E1 0000A     BBC     #1, COPY$SEM_STATUS+2, 2$               ;
                                 51  0000' CF  9E 00010 1$:     MOVAB   P.AAC, PROMPT_STRING_DESC               ; 1354
                                             05  11 00015     BRB     3$                                      ;
                                 51  0000' CF  9E 00017 2$:     MOVAB   P.AAE, PROMPT_STRING_DESC               ; 1355
                                 6E  0000G CF  9E 0001C 3$:     MOVAB   IN_NAME_DESC, PROMPT_ARGS               ; 1360
```

```
                04   AE      08   AE   9E 00021          MOVAB   OUT_DESC, PROMPT_ARGS+4                ; 1361
                50       0000G    CF   9A 00026          MOVZBL  OUTFILE_NAM_BLK+3, R0                  ; 1363
                                  0C   13 0002B          BEQL    4$                                     ;
                C3   AE           50   D0 0002D          MOVL    R0, OUT_DESC                           ; 1366
                0C   AE      0000G    CF   9E 00031          MOVAB   OUTFILE_NAME, OUT_DESC+4               ; 1367
                                  19   11 00037          BRB     6$                                     ; 1363
                50       0000G    CF   9A 00039 4$:      MOVZBL  OUTFILE_NAM_BLK+11, R0                 ; 1370
                                  0C   13 0003E          BEQL    5$                                     ;
                08   AE           50   D0 00040          MOVL    R0, OUT_DESC                           ; 1373
                0C   AE      0000G    CF   9E 00044          MOVAB   OUTFILE_XNAME, OUT_DESC+4              ; 1374
                                  06   11 0004A          BRB     6$                                     ; 1370
                04   AE      0000G    CF   9E 0004C 5$:      MOVAB   OUT_NAME_DESC, PROMPT_ARGS+4           ; 1377
                                  7E   D4 00052 6$:      CLRL    -(SP)                                  ; 1383
                           04   AE   9F 00054          PUSHAB  PROMPT_ARGS                            ;
                                  51   DD 00057          PUSHL   PROMPT_STRING_DESC                     ; 1384
                                  7E   D4 00059          CLRL    -(SP)                                  ; 1383
                           0000G    CF   9F 0005B          PUSHAB  INFILE_FAB                             ;
                           0000G    CF   9F 0005F          PUSHAB  COMMON_QUAL_CONTEXT                    ;
           00000000G   00          06   FB 00063          CALLS   #6, LIB$QUAL_FILE_MATCH                ;
                                  04   0006A          RET                                            ; 1386

; Routine Size:  107 bytes,    Routine Base:  $CODE$ + 0206
```

```
;  859          1387  1 ROUTINE CREATE_DIR (input_fab, output_fab) =
;  860          1388  1
;  861          1389  1 !---
;  862          1390  1 !
;  863          1391  1 !       This routine is called to create a directory file on
;  864          1392  1 !       the output side if the directory does not already exist.
;  865          1393  1 !       If the directory already exists, do nothing.
;  866          1394  1 !
;  867          1395  1 ! Inputs:
;  868          1396  1 !
;  869          1397  1 !       input_fab = Address of FAB describing opened directory file
;  870          1398  1 !       output_fab = Address of FAB describing the device and directory
;  871          1399  1 !                    into which the directory file should be created.
;  872          1400  1 !
;  873          1401  1 ! Outputs:
;  874          1402  1 !
;  875          1403  1 !       Routine value = status return
;  876          1404  1 !---
;  877          1405  1
;  878          1406  2 BEGIN
;  879          1407  2
;  880          1408  2 MAP
;  881          1409  2     input_fab:  REF BLOCK[,BYTE],        ! Input FAB
;  882          1410  2     output_fab: REF BLOCK[,BYTE];        ! Output FAB
;  883          1411  2
;  884          1412  2 BIND
;  885          1413  2     input_nam = .input_fab [fab$l_nam]: BLOCK[,BYTE],
;  886          1414  2     output_nam = .output_fab [fab$l_nam]: BLOCK[,BYTE];
;  887          1415  2
;  888          1416  2 LOCAL
;  889          1417  2     ptr,                                ! String temporary pointer
;  890          1418  2     addr,size,                          ! descriptor of search string
;  891          1419  2     buffer:     VECTOR [nam$c_maxrss,BYTE], ! file spec buffer
;  892          1420  2     bufdesc:    VECTOR [2],             ! descriptor of above buffer
;  893          1421  2     terminator: BYTE,                   ! Directory spec. terminator
;  894          1422  2     status;                             ! status variable
;  895          1423  2
;  896          1424  2 record_count = 0;                       ! Initialize the record count
;  897          1425  2 block_count  = 0;                       ! Initialize the block count
;  898          1426  2
;  899          1427  2 status = $RMS_PARSE (FAB = .output_fab); ! Get full name of directory file
;  900          1428  2
;  901          1429  2 size = .output_nam [nam$b_esl];         ! Get output expanded name
;  902          1430  2 addr = .output_nam [nam$l_esa];
;  903          1431  2
;  904          1432  2 IF NOT .status
;  905          1433  2 THEN
;  906          1434  3     BEGIN
;  907          1435  3     put_messagex(.status);
;  908          1436  3     RETURN .status;
;  909          1437  3     END;
;  910          1438  2
;  911          1439  2 ptr = CH$FIND_CH(.size, .addr, ']');    ! Find end of directory spec
;  912          1440  2 IF .ptr EQL 0                           ! If not found,
;  913          1441  2 THEN
;  914          1442  3     BEGIN
;  915          1443  3     ptr = CH$FIND_CH(.size, .addr, '>');   ! Alternate syntax
```

```
;  916          1444  3      IF .ptr EQL 0                           ! If still not found,
;  917          1445  3      THEN
;  918          1446  3          put_message(rms$_esa);               ! return invalid expanded string
;  919          1447  3      END;
;  920          1448  3
;  921          1449  3  size = .ptr + 1 - .addr;                    ! Figure length of device and dir.
;  922          1450  3  CH$MOVE(.size, .addr, buffer);              ! Copy device and directory into buffer
;  923          1451  3  terminator = .buffer [.size-1];             ! Remember terminator on dir. spec.
;  924          1452  3  buffer [.size-1] = '.';                     ! and overwrite it with '.'
;  925          1453  3
;  926          1454  3  bufdesc [0] = .size;                        ! Setup buffer descriptor
;  927          1455  3  bufdesc [1] = buffer;
;  928          1456  3
;  929          1457  3  size = .input_nam [nam$b_rsl];              ! Get input result name
;  930          1458  3  addr = .input_nam [nam$l_rsa];
;  931          1459  3
;  932          1460  3  ptr = CH$FIND_CH(.size, .addr, ']');       ! Find start of file name on input side
;  933          1461  3  IF .ptr EQL 0                              ! If not found,
;  934          1462  3  THEN
;  935          1463  3      BEGIN
;  936          1464  3      ptr = CH$FIND_CH(.size, .addr, '>'); ! Alternate syntax
;  937          1465  3      IF .ptr EQL 0                           ! If still not found
;  938          1466  3      THEN
;  939          1467  3          put_message(rms$_esa);               ! return invalid expanded string
;  940          1468  3      END;
;  941          1469  3
;  942          1470  3  size = .size - (.ptr + 1 - .addr);         ! Figure descriptor of file name
;  943          1471  3  addr = .ptr + 1;
;  944          1472  3
;  945          1473  3  ptr = CH$FIND_CH(.size, .addr, '.');       ! Find where file name ends
;  946          1474  3  IF .ptr EQL 0                              ! If not found,
;  947          1475  3  THEN
;  948          1476  3      RETURN rms$_esa;                         ! return invalid expanded string
;  949          1477  3  size = .ptr - .addr;                       ! Figure descriptor of file name only
;  950          1478  3
;  951          1479  3  CH$MOVE(.size, .addr, buffer+.bufdesc[0]); ! Append subdirectory name to buffer
;  952          1480  3  buffer [.bufdesc[0]+.size] = .terminator;  ! Tack terminator on end of it
;  953          1481  3  bufdesc [0] = .bufdesc[0] + .size + 1;  ! Update string descriptor
;  954          1482  3
;  955          1483  3  out_name_desc [0] = .bufdesc [0];          ! Copy length of string
;  956          1484  3  CH$MOVE(.bufdesc[0], .bufdesc[1], .out_name_desc[1]);   ! and string too
;  957          1485  3
;  958          1486  3  status = LIB$CREATE_DIR (bufdesc);         ! Create directory file with defaults
;  959          1487  3
;  960          1488  3  IF NOT .status                              ! If error detected,
;  961          1489  3  THEN
;  962          1490  3      put_messagex(.status);                   ! then signal status
;  963          1491  3
;  964          1492  3  RETURN .status;                             ! return with status
;  965          1493  3
;  966          1494  1  END;


                                                                    .EXTRN  SYS$PARSE

                                                    0FFC 00000 CREATE_DIR:
```

```
                                                              .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1387
                    5E      FEF8    CE  9E 00002              MOVAB   -264(SP), SP
                    50        04    AC  D0 00007              MOVL    INPUT_FAB, R0                                 ; 1413
                    58        28    A0  D0 0000B              MOVL    40(R0), R8
                    50        08    AC  D0 0000F              MOVL    OUTPUT_FAB, R0                                ; 1414
                    52        28    A0  D0 00013              MOVL    40(R0), R2
                           0000'    CF  7C 00017              CLRQ    BLOCK_COUNT                                   ; 1425
                                    50  DD 0001B              PUSHL   R0                                            ; 1427
         00000000G  00              01  FB 0001D              CALLS   #1, SYS$PARSE
                    5A              50  D0 00024              MOVL    R0, STATUS
                    57        0B    A2  9A 00027              MOVZBL  11(R2), SIZE                                  ; 1429
                    56        0C    A2  D0 0002B              MOVL    12(R2), ADDR                                  ; 1430
                    03              5A  E8 0002F              BLBS    STATUS, 1$                                    ; 1432
                                  00DD  31 00032              BRW     10$
      66            57        5D    8F  3A 00035  1$:         LOCC    #93, SIZE, (ADDR)                             ; 1439
                                    02  12 0003A              BNEQ    2$
                                    51  D4 0003C              CLRL    R1
                    59              51  D0 0003E  2$:         MOVL    R1, PTR
                                    21  12 00041              BNEQ    4$                                            ; 1440
      66            57              3E  3A 00043              LOCC    #62, SIZE, (ADDR)                             ; 1443
                                    02  12 00047              BNEQ    3$
                                    51  D4 00049              CLRL    R1
                    59              51  D0 0004B  3$:         MOVL    R1, PTR
                                    14  12 0004E              BNEQ    4$                                            ; 1444
                         000184FC   8F  DD 00050              PUSHL   #99580                                        ; 1446
             0000V  CF              01  FB 00056              CALLS   #1, COPY$MSG_NUMBER
                                    50  DD 0005B              PUSHL   R0
         00000000G  00              01  FB 0005D              CALLS   #1, LIB$STOP
      51            59              56  C3 00064  4$:         SUBL3   ADDR, PTR, R1                                 ; 1449
                    57        01    A1  9E 00068              MOVAB   1(R1), SIZE
  08  AE            66              57  28 0006C              MOVC3   SIZE, (ADDR), BUFFER                          ; 1450
                    5B        07 AE47   90 00071              MOVB    BUFFER-1[SIZE], TERMINATOR                    ; 1451
             07 AE47                2E  90 00076              MOVB    #46, BUFFER-1[SIZE]                           ; 1452
                    6E              57  D0 0007B              MOVL    SIZE, BUFDESC                                 ; 1454
             04     AE        08    AE  9E 0007E              MOVAB   BUFFER, BUFDESC+4                             ; 1455
                    57        03    A8  9A 00083              MOVZBL  3(R8), SIZE                                   ; 1457
                    56        04    A8  D0 00087              MOVL    4(R8), ADDR                                   ; 1458
      66            57        5D    8F  3A 0008B              LOCC    #93, SIZE, (ADDR)                             ; 1460
                                    02  12 00090              BNEQ    5$
                                    51  D4 00092              CLRL    R1
                    59              51  D0 00094  5$:         MOVL    R1, PTR
                                    21  12 00097              BNEQ    7$                                            ; 1461
      66            57              3E  3A 00099              LOCC    #62, SIZE, (ADDR)                             ; 1464
                                    02  12 0009D              BNEQ    6$
                                    51  D4 0009F              CLRL    R1
                    59              51  D0 000A1  6$:         MOVL    R1, PTR
                                    14  12 000A4              BNEQ    7$                                            ; 1465
                         000184FC   8F  DD 000A6              PUSHL   #99580                                        ; 1467
             0000V  CF              01  FB 000AC              CALLS   #1, COPY$MSG_NUMBER
                                    50  DD 000B1              PUSHL   R0
         00000000G  00              01  FB 000B3              CALLS   #1, LIB$STOP
      50            56              59  C3 000BA  7$:         SUBL3   PTR, ADDR, R0                                 ; 1470
                    57        FF A047   9E 000BE              MOVAB   -1(R0)[SIZE], SIZE
                    56        01    A9  9E 000C3              MOVAB   1(R9), ADDR                                   ; 1471
      66            57              2E  3A 000C7              LOCC    #46, SIZE, (ADDR)                             ; 1473
                                    02  12 000CB              BNEQ    8$
                                    51  D4 000CD              CLRL    R1
```

```
                          59            51 D0 000CF 8$:     MOVL    R1, PTR                                  ;
                                        08 12 000D2         BNEQ    9$                                       ; 1474
                       50 000184FC      8F D0 000D4         MOVL    #99580, R0                               ; 1476
                                        04 000DB            RET                                              ;
             57           59            56 C3 000DC 9$:     SUBL3   ADDR, PTR, SIZE                          ; 1477
                          50      08    AE 9E 000E0         MOVAB   BUFFER, R0                               ; 1479
      00 BE40             66            57 28 000E4         MOVC3   SIZE, (ADDR), @BUFDESC[R0]               ;
             50           6E            57 C1 000EA         ADDL3   SIZE, BUFDESC, R0                        ; 1480
                  08 AE40               5B 90 000EE         MOVB    TERMINATOR, BUFFER[R0]                   ;
                          6E      01    A0 9E 000F3         MOVAB   1(R0), BUFDESC                           ; 1481
                    0000G CF            6E D0 000F7         MOVL    BUFDESC, OUT_NAME_DESC                   ; 1483
    0000G  DF          04 BE            6E 28 000FC         MOVC3   BUFDESC, @BUFDESC+4, @OUT_NAME_DESC+4    ; 1484
                                        5E DD 00103         PUSHL   SP                                       ; 1486
                000000000G 00           01 FB 00105         CALLS   #1, LIB$CREATE_DIR                       ;
                          5A            50 D0 0010C         MOVL    R0, STATUS                               ;
                          38            5A E8 0010F         BLBS    STATUS, 12$                              ; 1488
                                        5A DD 00112 10$:    PUSHL   STATUS                                   ; 1490
                    0000V CF            01 FB 00114         CALLS   #1, COPY$MSG_NUMBER                      ;
    7E           00       50            01 7A 00119         EMUL    #1, R0, #0, -(SP)                        ;
    50           50       8E            08 7B 0011E         EDIV    #8, (SP)+, R0, R0                        ;
                          04            50 D1 00123         CMPL    R0, #4                                   ;
                                        12 13 00126         BEQL    11$                                      ;
                                        5A DD 00128         PUSHL   STATUS                                   ;
                    0000V CF            01 FB 0012A         CALLS   #1, COPY$MSG_NUMBER                      ;
                                        50 DD 0012F         PUSHL   R0                                       ;
                000000000G 00           01 FB 00131         CALLS   #1, LIB$SIGNAL                           ;
                                        10 11 00138         BRB     12$                                      ;
                                        5A DD 0013A 11$:    PUSHL   STATUS                                   ;
                    0000V CF            01 FB 0013C         CALLS   #1, COPY$MSG_NUMBER                      ;
                                        50 DD 00141         PUSHL   R0                                       ;
                000000000G 00           01 FB 00143         CALLS   #1, LIB$STOP                             ;
                          50            5A D0 0014A 12$:    MOVL    STATUS, R0                               ; 1492
                                        04 0014D            RET                                              ; 1494

; Routine Size:  334 bytes,    Routine Base:  $CODE$ + 0271
```

```
;  968          1495  1 ROUTINE RMS_SETUP =                                  ! RMS RAB setup routine
;  969          1496  1
;  970          1497  1 !++
;  971          1498  1 ! FUNCTIONAL DESCRIPTION:
;  972          1499  1 !
;  973          1500  1 !       This routine performs all necessary setup of the input and output file RABs:
;  974          1501  1 !
;  975          1502  1 !               * determine if record-mode is required
;  976          1503  1 !               * allocate I/O buffers
;  977          1504  1 !               * connect the RABs to their respective FABs
;  978          1505  1 !
;  979          1506  1 ! FORMAL PARAMETERS:
;  980          1507  1 !
;  981          1508  1 !       None
;  982          1509  1 !
;  983          1510  1 ! IMPLICIT INPUTS:
;  984          1511  1 !
;  985          1512  1 !       EXTEND_OUTFILE - Indicates output file is being extended
;  986          1513  1 !       IO_BUFFER_BASE - location of the I/O buffer pool
;  987          1514  1 !       INFILE_FAB - Input file FAB
;  988          1515  1 !       OUTFILE_FAB - Output file FAB
;  989          1516  1 !       INFILE_XABs - Input file XABs
;  990          1517  1 !
;  991          1518  1 ! IMPLICIT OUTPUTS:
;  992          1519  1 !
;  993          1520  1 !       INFILE_RAB - Input file RAB completed and connected
;  994          1521  1 !       OUTFILE_RAB - Output file RAB completed and connected
;  995          1522  1 !       IO_BUFFER_BASE - Address of dynamic I/O buffer (1st call only)
;  996          1523  1 !       BLOCK_IO_SIZE - Length of block I/O operations
;  997          1524  1 !
;  998          1525  1 ! COMPLETION CODES:
;  999          1526  1 !
; 1000          1527  1 !       OK = normal completion
; 1001          1528  1 !       ERROR = RAB connect unsuccessful
; 1002          1529  1 !
; 1003          1530  1 ! SIDE EFFECTS:
; 1004          1531  1 !
; 1005          1532  1 !       None
; 1006          1533  1 !
; 1007          1534  1 !--
; 1008          1535  1
; 1009          1536  2     BEGIN
; 1010          1537  2
; 1011          1538  2     LOCAL
; 1012          1539  2         IN_DEVICE :  BLOCK[1,BYTE],                ! Selected input and output
; 1013          1540  2         OUT_DEVICE :  BLOCK[1,BYTE],               ! device characteristics
; 1014          1541  2
; 1015          1542  2         FORCE_REC_MODE,                            ! Temporary record-mode I/O indicator
; 1016          1543  2         STATUS,                                    ! System service completion code
; 1017          1544  2         IO_BUFFER_LENGTH : INITIAL(max_io_length*2),    ! Size of I/O buffer pool
; 1018          1545  2         GETSYI_ITEM_LIST : $ITMLST_DECL(ITEMS=1);  ! Item list for $GETSYI call
; 1019          1546  2
; 1020          1547  2     MACRO                                          ! IN_DEVICE and OUT_DEVICE bit definitions:
; 1021          1548  2         DISK = 0,0,1,0 %,                          !    disk device
; 1022          1549  2         TAPE = 0,1,1,0 %;                          !    tape device
; 1023          1550  2
; 1024          1551  2 !
```

```
;  1025     1552  2 ! Allocate a maximum size I/O buffer pool on the 1st call to this routine.
;  1026     1553  2 !
;  1027     1554  2
;  1028     1555  2     IF .io_buffer_base EQL 0
;  1029     1556  2     THEN
;  1030     1557  2         BEGIN
;  1031     1558  2
;  1032     1559  2         !
;  1033     1560  2         ! Allocate enough virtual memory for the I/O buffer pool.  It has to
;  1034     1561  2         ! be large enough to hold two of the largest possible RMS transfers.
;  1035     1562  2         ! ****** NOTE ****** If COPY is ever made callable, the allocation of
;  1036     1563  2         ! the I/O buffer pool will have to be rewritten to be more efficient.
;  1037     1564  2         !
;  1038     1565  2
;  1039     1566  2         IF NOT (status = LIB$GET_VM (io_buffer_length, io_buffer_base))
;  1040     1567  2         THEN
;  1041     1568  2             PUT_MESSAGE( MSG$_BADLOGIC, 0, .STATUS, 0,  MSG$_ATPC, 1 );
;  1042     1569  2         END;
;  1043     1570  2
;  1044     1571  2 !
;  1045     1572  2 ! Extract some device information from the input and output file FABs.
;  1046     1573  2 !
;  1047     1574  2
;  1048     1575  2     IN_DEVICE = 0;                                  ! Clear the input and output
;  1049     1576  2     OUT_DEVICE = 0;                                 ! device characteristics.
;  1050     1577  2
;  1051     1578  2     IN_DEVICE[DISK] =                               ! Turn on the input file disk indicator
;  1052     1579  2         .INFILE_FAB[$FAB_DEV(FOD)] AND              ! if the input device is file-structured
;  1053     1580  2         NOT .INFILE_FAB[$FAB_DEV(SQD)];             ! and it is not a tape device.
;  1054     1581  2
;  1055     1582  2     IN_DEVICE[TAPE] =                               ! Turn on the input file tape indicator
;  1056     1583  2         .INFILE_FAB[$FAB_DEV(SQD)];                 ! if the input device is a tape.
;  1057     1584  2
;  1058     1585  2     OUT_DEVICE[DISK] =                              ! Turn on the output file disk indicator
;  1059     1586  2         .OUTFILE_FAB[$FAB_DEV(FOD)] AND             ! if the output device is file-structured
;  1060     1587  2         NOT .OUTFILE_FAB[$FAB_DEV(SQD)];            ! and it is not a tape device.
;  1061     1588  2
;  1062     1589  2     OUT_DEVICE[TAPE] =                              ! Turn on the output file tape indicator
;  1063     1590  2         .OUTFILE_FAB[$FAB_DEV(SQD)];                ! if the output device is a tape.
;  1064     1591  2
;  1065     1592  2 !
;  1066     1593  2 ! Determine whether the input and output files have compatible attributes. This
;  1067     1594  2 ! check can only be done if both the input and output devices are the same kind
;  1068     1595  2 ! and they are file structured. The check should not be done if either the
;  1069     1596  2 ! input device or the output device is a network device.
;  1070     1597  2 !
;  1071     1598  2
;  1072     1599  2     IF .in_device NEQ .out_device
;  1073     1600  2                 OR
;  1074     1601  2        .in_device EQL 0
;  1075     1602  2
;  1076     1603  2     THEN
;  1077     1604  2         force_rec_mode = YES
;  1078     1605  2     ELSE
;  1079     1606  2         IF NOT(.infile_fab[$FAB_DEV(NET)]           ! If neither input or output is network then
;  1080     1607  2                 OR
;  1081     1608  3             .outfile_fab[$FAB_DEV(NET)])
```

```
                AND                                          ! Compare the following input and output XAB fields:
            (IN_NEQ_OUT(XAB$B_RFO) OR                        !    record format and file organization
             IN_NEQ_OUT(XAB$B_ATR) OR                        !    record attributes
             IN_NEQ_OUT(XAB$B_BKZ) OR                        !    bucket size
             IN_NEQ_OUT(XAB$B_HSZ) OR                        !    fixed header size
             (.OUTFILE_XABFHC[XAB$W_MRZ] NEQ 0 AND           !    maximum output record size (if any)
              .OUTFILE_XABFHC[XAB$W_MRZ] LSS                 !       and longest input record
                    .INFILE_XABFHC[XAB$W_LRL]))              !
        THEN
            BEGIN                                            ! If the input and output attributes are not identic
            IF NOT .COPY$B_INCOMPAT                          ! and this message has not appeared yet
            THEN                                             ! for this output file.
                BEGIN
                PUT_MESSAGE( MSG$_INCOMPAT, 2,               ! send the user a warning message
                        IN_NAME_DESC, OUT_NAME_DESC );       !
                COPY$B_INCOMPAT = TRUE;                      ! Set flag saying that message is out.
                END;
            FORCE_REC_MODE = YES;                            ! and force a record-mode copy.
            END
        ELSE
            FORCE_REC_MODE = NO;                             ! Otherwise, turn the record-mode indicator off.

    !
    ! Initialize the input and output RABs.
    !

    $RAB_INIT( RAB = INFILE_RAB,                             ! Setup the input file RAB as follows:
               RAC = SEQ,                                    !    Sequential record access
               ROP = <LOC,RAH>,                              !    GET locate, read ahead
               FAB = INFILE_FAB);                            !    Input file FAB address

    $RAB_INIT( RAB = OUTFILE_RAB,                            ! Setup the output file RAB as follows:
               RAC = SEQ,                                    !    Sequential access
               FAB = OUTFILE_FAB,                            !    Output file FAB address
               ROP = <TPT,WBH> );                            !    Force EOF on every write or put,
                                                             !    and specify write behind for multi-buffering.

    !
    !
    ! Determine whether record-mode I/O is required for this file copy operation.
    ! At least one of the following conditions must be true for record mode
    ! operations to be performed:
    !         - the input and output attributes are incompatible,
    !         - the output file is being extended,
    !         - the input and output devices are not the same type,
    !         - both devices are record mode devices,
    !         - this is a tape-to-tape copy AND
    !             the input and output blocksizes are not the same
    !                     OR
    !             one tape is mounted foreign and the other is ANSI.
    !

    IF .FORCE_REC_MODE
            OR
        .EXTEND_OUTFILE
            OR
        .IN_DEVICE NEQ .OUT_DEVICE
```

```
2              OR
2          .IN_DEVICE EQL 0
2              OR
3          (
3          .INFILE_FAB [$FAB_DEV (SQD)]
3              AND
4          (
5              ( .INFILE_FAB [FAB$W_BLS] NEQ .OUTFILE_FAB [FAB$W_BLS] )
4              OR
5              ( .INFILE_FAB [$FAB_DEV (FOR)] NEQ .OUTFILE_FAB [$FAB_DEV (FC )] )
4          )
3          )
3  !
3  ! Record mode I/O setup.
3  !
3
2      THEN
3          BEGIN
3
3          !
3          ! Indicate that record mode is required, block i/o will not be used for
3          ! this file, and that the record operations will be synchronous.
3          !
3          record_mode = YES;
3          infile_rab[RAB$V_BIO]  = NO;
3          outfile_rab[RAB$V_BIO] = NO;
3          infile_rab[RAB$V_ASY]  = NO;
3          outfile_rab[RAB$V_ASY] = NO;
3
3          !
3          ! Determine the size of the user's buffer which is passed to RMS.
3          ! If the input device is tape, then the user's buffer must be large
3          ! enough to contain one complete tape block.  Otherwise, (the input
3          ! device is not tape) use either the the maximum record size or the
3          ! the longest record length for the size of the user's buffer, if they
3          ! are specfied.  If none of the above cases are met, use the longest
3          ! legal transfer size as the length of the user's buffer.
3          !
3          IF .in_device[tape]
3          THEN
3              infile_rab[RAB$W_USZ] = .infile_fab[FAB$W_BLS]
3          ELSE
3              IF .infile_xabfhc[XAB$W_MRZ] NEQ 0
3              THEN
3                  infile_rab[RAB$W_USZ] = .infile_xabfhc[XAB$W_MRZ]
3              ELSE
3                  IF .infile_xabfhc[XAB$W_LRL] NEQ 0
3                  THEN
3                      infile_rab[RAB$W_USZ] = .infile_xabfhc[XAB$W_LRL]
3                  ELSE
3                      infile_rab[RAB$W_USZ] = max_io_length;
3
3
3          !
3          ! Set up the user's buffer within the I/O buffer pool.  If the record
3          ! format of the file is VFC, then allocate areas in the buffer pool
3          ! for the fixed header and variable portions of the record.  Otherwise,
```

```
; 1196          1723  3          ! just use the start of the I/O buffer pool as the start of the user's
; 1197          1724  3          ! buffer.
; 1198          1725  3          !
; 1199          1726  3          IF .infile_fab[FAB$B_RFM] EQL FAB$C_VFC
; 1200          1727  3          THEN
; 1201          1728  4              BEGIN
; 1202          1729  4              infile_rab[RAB$L_RHB]  = .io_buffer_base;
; 1203          1730  4              outfile_rab[RAB$L_RHB] = .infile_rab[RAB$L_RHB];
; 1204          1731  4              infile_rab[RAB$L_OBF]  = .io_buffer_base + .infile_xabfhc[XAB$B_HSZ];
; 1205          1732  4              END
; 1206          1733  3          ELSE
; 1207          1734  3              infile_rab[RAB$L_UBF] = .io_buffer_base;
; 1208          1735  3          !
; 1209          1736  3          !
; 1210          1737  3          ! Determine the best multi-block count for copying the input file.  Use
; 1211          1738  3          ! that MBC for both the input and output file RABs.
; 1212          1739  3          !
; 1213          1740  3          IF .infile_fab [FAB$W_BLS] GTR .outfile_fab [FAB$W_BLS]
; 1214          1741  3          THEN
; 1215          1742  3
; 1216          1743  3              ! The input device is tape or some other record oriented device.
; 1217          1744  3              ! Have RMS allocate enough buffer space to hold a complete block.
; 1218          1745  3              !
; 1219          1746  3              infile_rab [RAB$B_MBC] = (.infile_fab [FAB$W_BLS] + 511) / disk_block_size
; 1220          1747  3          ELSE
; 1221          1748  3              IF .outfile_fab [FAB$W_BLS] NEQ 0
; 1222          1749  3              THEN
; 1223          1750  3
; 1224          1751  3                  ! The output device is record oriented and its block size is
; 1225          1752  3                  ! larger than the input device's.  Therefore, RMS should
; 1226          1753  3                  ! allocate enough buffer space to hold a comlete block for the
; 1227          1754  3                  ! output device.
; 1228          1755  3                  !
; 1229          1756  3                  infile_rab [RAB$B_MBC] = (.outfile_fab [FAB$W_BLS] + 511) / disk_block_size
; 1230          1757  3              ELSE
; 1231          1758  3
; 1232          1759  3                  ! This is either a disk to disk transfer or something else.
; 1233          1760  3                  ! Just use the system default.
; 1234          1761  3                  !
; 1235          1762  3                  infile_rab [RAB$B_MBC] = .rms_mbc;
; 1236          1763  3
; 1237          1764  3          outfile_rab [RAB$B_MBC] = .infile_rab [RAB$B_MBC];
; 1238          1765  3
; 1239          1766  3          !
; 1240          1767  3          ! Have RMS set up two internal buffers, to speed up processing.
; 1241          1768  3          !
; 1242          1769  3          infile_rab [RAB$B_MBF]  = double_buffer;
; 1243          1770  3          outfile_rab [RAB$B_MBF] = double_buffer;
; 1244          1771  3          END
; 1245          1772  3
; 1246          1773  3
; 1247          1774  3      ! Block mode I/O setup.
; 1248          1775  3      !
; 1249          1776  3
; 1250          1777  2      ELSE
; 1251          1778  3          BEGIN
; 1252          1779  3
```

```
;  1253          1780  3          !
;  1254          1781  3          ! Indicate that record mode is not desired and that block mode will be
;  1255          1782  3          ! used for both input and output, and that reading and writing will be
;  1256          1783  3          ! synchronous.  However, ASY will be set after the $CONNECT to avoid
;  1257          1784  3          ! haveing to issue a $WAIT on the connect.
;  1258          1785  3          !
;  1259          1786  3
;  1260          1787  3          record_mode = NO;
;  1261          1788  3          infile_rab[RAB$V_BIO] = YES;
;  1262          1789  3          outfile_rab[RAB$V_BIO] = YES;
;  1263          1790  3          infile_rab[RAB$V_ASY] = NO;
;  1264          1791  3          outfile_rab[RAB$V_ASY] = NO;
;  1265          1792  3
;  1266          1793  3          !
;  1267          1794  3          ! Determine the appropriate block size and user buffer size for copying
;  1268          1795  3          ! the current input file.
;  1269          1796  3          !
;  1270          1797  3          IF .in_device[tape]
;  1271          1798  3          THEN
;  1272          1799  4              BEGIN
;  1273          1800  4              block_size = .infile_fab [FAB$W_BLS];
;  1274          1801  4              infile_rab[RAB$W_USZ] = .infile_fab[FAB$W_BLS];
;  1275          1802  4              END
;  1276          1803  3          ELSE
;  1277          1804  4              BEGIN
;  1278          1805  4              block_size = disk_block_size;
;  1279          1806  4              infile_rab[RAB$W_USZ] = .rms_mbc * disk_block_size;
;  1280          1807  3              END;
;  1281          1808  3
;  1282          1809  3          !
;  1283          1810  3          ! Set up the user's buffer, which are passed to RMS, within the I/O
;  1284          1811  3          ! buffer pool.
;  1285          1812  3          !
;  1286          1813  3
;  1287          1814  3          infile_rab[RAB$L_UBF]  = .io_buffer_base;
;  1288          1815  3          outfile_rab[RAB$[_RBF] = .io_buffer_base + .infile_rab[RAB$W_USZ];
;  1289          1816  3          END;
;  1290          1817  2
;  1291          1818  2      !
;  1292          1819  2      ! Connect the input and output RABs to their respective FABs.
;  1293          1820  2      !
;  1294          1821  2
;  1295        P 1822  2          IF NOT $RMS_CONNECT( RAB = INFILE_RAB,                  ! Connect the input file RAB to the FAB,
;  1296          1823  2                               ERR = COPY$INOPN_ERR )             ! specifying an error action routine.
;  1297          1824  2
;  1298          1825  2          THEN                                                    ! If the connect was not successful,
;  1299          1826  2              RETURN NO_FILE;                                     ! return an error indication to the caller.
;  1300          1827  2
;  1301          1828  2          IF .EXTEND_OUTFILE                                      ! If the output file is being extended,
;  1302          1829  2          THEN                                                    !
;  1303          1830  2              OUTFILE_RAB[RAB$V_EOF] = YES;                       ! force end-of-file positioning on the following CON
;  1304          1831  2
;  1305        P 1832  2          IF NOT $RMS_CONNECT( RAB = OUTFILE_RAB,                 ! Connect the output file RAB to the FAB,
;  1306          1833  2                               ERR = COPY$OUTOPN_ERR )            ! specifying an error action routine.
;  1307          1834  2
;  1308          1835  2          THEN                                                    ! If the connect was not successful,
;  1309          1836  2              RETURN NO_FILE;                                     ! return an error indication to the caller.
```

```
2
2 !
2 ! Set ASY bit in ROP if block I/O mode.
2 !
2
2     IF NOT .RECORD_MODE                                                ! If block I/O mode
2     THEN                                                               !
3         BEGIN                                                          !
3         INFILE_RAB[RAB$V_ASY] = YES;                                   ! indicate that reading and
3         OUTFILE_RAB[RAB$V_ASY] = YES;                                  ! writing will be asynchronous
2         END;                                                           !
2
2 !
2 ! Return to the caller
2 !
2
2     RETURN OK;                                                         ! Return a success code to the caller.
1     END;


                                                         .EXTRN  SYS$CONNECT

                                    0FFC 00000 RMS_SETUP:
                                                         .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11     ; 1495
                    5B      0000G  CF  9E 00002          MOVAB   INFILE_FAB+64, R11                        ;
                    5A      0000G  CF  9E 00007          MOVAB   $RMS_PTR+4, R10                           ;
                    59      0000G  CF  9E 0000C          MOVAB   $RMS_PTR+4, R9                            ;
                    5E             10  C2 00011          SUBL2   #16, SP                                   ;
                       0001FFFE    8F  DD 00014          PUSHL   #131070                                   ; 1536
                            0000'  CF  D5 0001A          TSTL    IO_BUFFER_BASE                            ; 1555
                                   31  12 0001E          BNEQ    1$                                        ;
                            0000'  CF  9F 00020          PUSHAB  IO_BUFFER_BASE                            ; 1566
                              04   AE  9F 00024          PUSHAB  IO_BUFFER_LENGTH                          ;
          00000000G 00             02  FB 00027          CALLS   #2, LIB$GET_VM                            ;
                    20             50  E8 0002E          BLBS    STATUS, 1$                                ;
                                   01  DD 00031          PUSHL   #1                                        ; 1568
                    7E      115A   8F  3C 00033          MOVZWL  #4442, -(SP)                              ;
                                   7E  D4 00038          CLRL    -(SP)                                     ;
                                   50  DD 0003A          PUSHL   STATUS                                    ;
                                   7E  D4 0003C          CLRL    -(SP)                                     ;
                    7E      1124   8F  3C 0003E          MOVZWL  #4388, -(SP)                              ;
              0000V CF             01  FB 00043          CALLS   #1, COPY$MSG_NUMBER                       ;
                                   50  DD 00048          PUSHL   R0                                        ;
          00000000G 00             06  FB 0004A          CALLS   #6, LIB$STOP                              ;
                                   57  94 00051 1$:      CLRB    IN_DEVICE                                 ; 1575
                                   50  94 00053          CLRB    OUT_DEVICE                                ; 1576
51      01   AB     01             06  EF 00055          EXTZV   #6, #1, INFILE_FAB+65, R1                 ; 1580
52           6B     01             05  EF 0005B          EXTZV   #5, #1, INFILE_FAB+64, R2                 ;
                    51             52  8A 00060          BICB2   R2, R1                                    ;
57           01     00             51  F0 00063          INSV    R1, #0, #1, IN_DEVICE                     ;
51           6B     01             05  EF 00068          EXTZV   #5, #1, INFILE_FAB+64, R1                 ; 1583
57           01     01             51  F0 0006D          INSV    R1, #1, #1, IN_DEVICE                     ;
51    0000G  CF     01             06  EF 00072          EXTZV   #6, #1, OUTFILE_FAB+65, R1                ; 1587
52    0000G  CF     01             05  EF 00079          EXTZV   #5, #1, OUTFILE_FAB+64, R2                ;
                    51             52  8A 00080          BICB2   R2, R1                                    ;
50           01     00             51  F0 00083          INSV    R1, #0, #1, OUT_DEVICE                    ;
```

```
                51      0000G   CF              01              05  EF  00088           EXTZV   #5, #1, OUTFILE_FAB+64, R1                      ; 1590
                50              01              01              51  F0  0008F           INSV    R1, #1, #1, OUT_DEVICE                          ;
                                                                58  D4  00094           CLRL    R8                                              ; 1599
                                                50              57  91  00096           CMPB    IN_DEVICE, OUT_DEVICE                           ;
                                                                04  13  00099           BEQL    2$                                              ;
                                                                58  D6  0009B           INCL    R8                                              ;
                                                                68  11  0009D           BRB     4$                                              ;
                                                                57  95  0009F 2$:       TSTB    IN_DEVICE                                       ; 1601
                                                                64  13  000A1           BEQL    4$                                              ;
                        64      01      AB                      05  E0  000A3           BBS     #5, INFILE_FAB+65, 5$                           ; 1606
                        5E      0000G   CF                      05  E0  000A8           BBS     #5, OUTFILE_FAB+65, 5$                          ; 1608
                                0000G   CF      0000G   CF      91  00AE              CMPB    INFILE_XABFHC+8, OUTFILE_XABFHC+8               ; 1610
                                                                29  12  000B5           BNEQ    3$                                              ;
                                0000G   CF      0000G   CF      91  000B7           CMPB    INFILE_XABFHC+9, OUTFILE_XABFHC+9               ; 1611
                                                                20  12  000BE           BNEQ    3$                                              ;
                                0000G   CF      0000G   CF      91  000C0           CMPB    INFILE_XABFHC+22, OUTFILE_XABFHC+22             ; 1612
                                                                17  12  000C7           BNEQ    3$                                              ;
                                0000G   CF      0000G   CF      91  000C9           CMPB    INFILE_XABFHC+23, OUTFILE_XABFHC+23             ; 1613
                                                                0E  12  000D0           BNEQ    3$                                              ;
                                        50      0000G   CF      3C  000D2           MOVZWL  OUTFILE_XABFHC+24, R0                           ; 1614
                                                                33  13  000D7           BEQL    5$                                              ;
                                        50      0000G   CF      B1  000D9           CMPW    INFILE_XABFHC+10, R0                            ; 1616
                                                                2C  1B  000DE           BLEQU   5$                                              ;
                                        22      0000'   CF      E8  000E0 3$:       BLBS    COPY$B_INCOMPAT, 4$                             ; 1619
                                                0000G   CF      9F  000E5           PUSHAB  OUT_NAME_DESC                                   ; 1623
                                                0000G   CF      9F  000E9           PUSHAB  IN_NAME_DESC                                    ;
                                                                02  DD  000ED           PUSHL   #2                                              ;
                                        7E      11E0            8F  3C  000EF           MOVZWL  #4576, -(SP)                                    ;
                                0000V   CF                      01  FB  000F4           CALLS   #1, COPY$MSG_NUMBER                             ;
                                                                50  DD  000F9           PUSHL   R0                                              ;
                        00000000G       00                      04  FB  000FB           CALLS   #4, LIB$SIGNAL                                  ;
                                  0000' CF                      01  90  00102           MOVB    #1, COPY$B_INCOMPAT                             ; 1624
                                        56                      01  D0  00107 4$:       MOVL    #1, FORCE_REC_MODE                              ; 1626
                                                                02  11  0010A           BRB     6$                                              ; 1606
                                                                56  D4  0010C 5$:       CLRL    FORCE_REC_MODE                                  ; 1629
        0044    8F              00              6E                      00  2C  0010E 6$:       MOVC5   #0, (SP), #0, #68, $RMS_PTR             ; 1638
                                                        FC              A9  00115
                                                FC      A9      4401    8F  B0  00117           MOVW    #17409, $RMS_PTR                        ;
                                                        69  00010200    8F  D0  0011D           MOVL    #66048, $RMS_PTR+4                      ;
                                                            1A          A9  94  00124           CLRB    $RMS_PTR+30                             ;
                                                3B      A9      C0      AB  9E  00127           MOVAB   INFILE_FAB, $RMS_PTR+60                 ;
        0044    8F              00              6E                      00  2C  0012C           MOVC5   #0, (SP), #0, #68, $RMS_PTR             ; 1643
                                                        FC              AA  00133
                                                FC      AA      4401    8F  B0  00135           MOVW    #17409, $RMS_PTR                        ;
                                                        6A      0402    8F  3C  0013B           MOVZWL  #1026, $RMS_PTR+4                       ;
                                                            1A          AA  94  00140           CLRB    $RMS_PTR+30                             ;
                                                3B      AA      0000G   CF  9E  00143           MOVAB   OUTFILE_FAB, $RMS_PTR+60                ;
                                                        51      0000'   CF  D0  00149           MOVL    IO_BUFFER_BASE, R1                      ; 1729
                                                        26              56  E8  0014E           BLBS    FORCE_REC_MODE, 9$                      ; 1661
                                                                0000'   CF  95  00151           TSTB    COPY$SEM_STATUS+2                       ; 1663
                                                                        20  19  00155           BLSS    9$                                      ;
                                                        1D              58  E8  00157           BLBS    R8, 9$                                  ; 1665
                                                                        57  95  0015A           TSTB    IN_DEVICE                               ; 1667
                                                                        19  13  0015C           BEQL    9$                                      ;
                                                03              6B              05  E0  0015E           BBS     #5, INFILE_FAB+64, 8$           ; 1670
                                                                    00B6    31  00162 7$:       BRW     19$                                     ;
                                                0000G   CF      FC      AB  B1  00165 8$:       CMPW    INFILE_FAB+60, OUTFILE_FAB+60           ; 1673
```

```
                                  0A  12 0016B        BNEQ    9$                                         ;
         50      0000G  CF    03  AB  8D 0016D        XORB3   INFILE_FAB+67, OUTFILE_FAB+67, R0          ; 1675
                        EB        50  E9 00174        BLBC    R0, 7$                                     ;
                 0000'  CF    40  8F  88 00177 9$:    BISB2   #64, COPY$SEM_STATUS+2                     ; 1689
                    01  A9        08  8A 0017D        BICB2   #8, INFILE_RAB+5                           ; 1690
                    01  AA        08  8A 00181        BICB2   #8, OUTFILE_RAB+5                          ; 1691
                        69        01  8A 00185        BICB2   #1, INFILE_RAB+4                           ; 1692
                        6A        01  8A 00188        BICB2   #1, OUTFILE_RAB+4                          ; 1693
         07             57        01  E1 0018B        BBC     #1, IN_DEVICE, 10$                         ; 1704
                 1C     A9    FC  AB  B0 0018F        MOVW    INFILE_FAB+60, INFILE_RAB+32               ; 1706
                                  18  11 00194        BRB     13$                                        ;
                        50 0000G  CF  3C 00196 10$:   MOVZWL  INFILE_XABFHC+24, R0                       ; 1708
                                  07  12 0019B        BNEQ    11$                                        ;
                        50 0000G  CF  3C 0019D        MOVZWL  INFILE_XABFHC+10, R0                       ; 1712
                                  06  13 001A2        BEQL    12$                                        ;
                 1C     A9        50  B0 001A4 11$:   MOVW    R0, INFILE_RAB+32                          ; 1714
                                  04  11 001A8        BRB     13$                                        ;
                 1C     A9        01  AE 001AA 12$:   MNEGW   #1, INFILE_RAB+32                          ; 1716
                        03    DF  AB  91 001AE 13$:   CMPB    INFILE_FAB+31, #3                          ; 1726
                                  15  12 001B2        BNEQ    14$                                        ;
                 28     A9        51  D0 001B4        MOVL    R1, INFILE_RAB+44                          ; 1729
                 28     AA    28  A9  D0 001B8        MOVL    INFILE_RAB+44, OUTFILE_RAB+44              ; 1730
                        50 0000G  CF  9A 001BD        MOVZBL  INFILE_XABFHC+23, R0                       ; 1731
  20     A9             51        50  C1 001C2        ADDL3   R0, R1, INFILE_RAB+36                      ;
                                  04  11 001C7        BRB     15$                                        ; 1726
                 20     A9        51  D0 001C9 14$:   MOVL    R1, INFILE_RAB+36                          ; 1734
                        50 0000G  CF  3C 001CD 15$:   MOVZWL  OUTFILE_FAB+60, R0                         ; 1740
                        50    FC  AB  B1 001D2        CMPW    INFILE_FAB+60, R0                          ;
                                  17  1B 001D6        BLEQU   16$                                        ;
                        51    FC  AB  3C 001D8        MOVZWL  INFILE_FAB+60, R1                          ; 1746
                        51  01FF  C1  9E 001DC        MOVAB   511(R1), R1                                ;
         52             51 00000200 8F C7 001E1       DIVL3   #512, R1, R2                               ;
                 33     A9        52  90 001E9        MOVB    R2, INFILE_RAB+55                          ;
                                  1D  11 001ED        BRB     18$                                        ;
                                  50  D5 001EF 16$:   TSTL    R0                                         ; 1748
                                  13  13 001F1        BEQL    17$                                        ;
                        50  01FF  C0  9E 001F3        MOVAB   511(R0), R0                                ; 1756
         51             50 00000200 8F C7 001F8       DIVL3   #512, R0, R1                               ;
                 33     A9        51  90 00200        MOVB    R1, INFILE_RAB+55                          ;
                                  06  11 00204        BRB     18$                                        ;
                 33     A9 0000'  CF  90 00206 17$:   MOVB    RMS_MBC, INFILE_RAB+55                     ; 1762
                 33     AA    33  A9  90 0020C 18$:   MOVB    INFILE_RAB+55, OUTFILE_RAB+55              ; 1764
                 32     A9        02  90 00211        MOVB    #2, INFILE_RAB+54                          ; 1769
                 32     AA        02  90 00215        MOVB    #2, OUTFILE_RAB+54                         ; 1770
                                  42  11 00219        BRB     22$                                        ; 1661
                 0000'  CF    40  8F  8A 0021B 19$:   BICB2   #64, COPY$SEM_STATUS+2                     ; 1787
                    01  A9        08  88 00221        BISB2   #8, INFILE_RAB+5                           ; 1788
                    01  AA        08  88 00225        BISB2   #8, OUTFILE_RAB+5                          ; 1789
                        69        01  8A 00229        BICB2   #1, INFILE_RAB+4                           ; 1790
                        6A        01  8A 0022C        BICB2   #1, OUTFILE_RAB+4                          ; 1791
         0D             57        01  E1 0022F        BBC     #1, IN_DEVICE, 20$                         ; 1797
                 0000'  CF    FC  AB  3C 00233        MOVZWL  INFILE_FAB+60, BLOCK_SIZE                  ; 1800
                    1C  A9    FC  AB  B0 00239        MOVW    INFILE_FAB+60, INFILE_RAB+32               ; 1801
                                  10  11 0023E        BRB     21$                                        ; 1797
                 0000'  CF  0200  8F  3C 00240 20$:   MOVZWL  #512, BLOCK_SIZE                           ; 1805
  1C     A9      0000'  CF  0200  8F  A5 00247        MULW3   #512, RMS_MBC, INFILE_RAB+32               ; 1806
                    20  A9        51  D0 00250 21$:   MOVL    R1, INFILE_RAB+36                          ; 1814
```

```
                        50      1C   A9  3C 00254          MOVZWL  INFILE_RAB+32, R0                       ; 1815
                    24  AA          51  C1  50 00258          ADDL3   R0, R1, OUTFILE_RAB+40
                                0000V   CF  9F 0025D 22$:     PUSHAB  COPY$INOPN_ERR                          ; 1823
                                  FC    A9  9F 00261          PUSHAB  INFILE_RAB
                        00000000G   00      02  FB 00264          CALLS   #2, SYS$CONNECT
                                    2B      50  E9 00268          BLBC    R0, 25$
                                0000'   CF  95 0026E          TSTB    COPY$SEM_STATUS+2                   ; 1828
                                            04  18 00272          BGEQ    23$
                            01  AA          01  88 00274          BISB2   #1, OUTFILE_RAB+5               ; 1830
                                0000V   CF  9F 00278 23$:     PUSHAB  COPY$OUTOPN_ERR                     ; 1833
                                  FC    AA  9F 0027C          PUSHAB  OUTFILE_RAB
                        00000000G   00      02  FB 0027F          CALLS   #2, SYS$CONNECT
                                    10      50  E9 00286          BLBC    R0, 25$
                    06      0000'   CF      06  E0 00289          BBS     #6, COPY$SEM_STATUS+2, 24$      ; 1842
                                    69      01  88 0028F          BISB2   #1, INFILE_RAB+4                ; 1845
                                    6A      01  88 00292          BISB2   #1, OUTFILE_RAB+4               ; 1846
                                    50      01  D0 00295 24$:     MOVL    #1, R0                          ; 1853
                                                04  00298          RET
                                            50  D4 00299 25$:     CLRL    R0                              ; 1854
                                                04  0029B          RET

; Routine Size:  668 bytes,    Routine Base:  $CODE$ + 03BF
```

```
;  1329          1855  1 PSECT CODE = COPY$COPY_FILE (ALIGN(9));                    ! Force page alignment for this routine.
;  1330          1856  1
;  1331          1857  1 ROUTINE COPY_FILE =                                       ! Copies an entire input file to the output file
;  1332          1858  1
;  1333          1859  1 !++
;  1334          1860  1 ! FUNCTIONAL DESCRIPTION:
;  1335          1861  1 !
;  1336          1862  1 !       This routine copies an entire input file into the output file,
;  1337          1863  1 !       using block mode I/O if possible.
;  1338          1864  1 !
;  1339          1865  1 !       This routine is page-aligned in order to minimize page faulting
;  1340          1866  1 !       due to executing the code which performs the actual file copying.
;  1341          1867  1 !
;  1342          1868  1 ! FORMAL PARAMETERS:
;  1343          1869  1 !
;  1344          1870  1 !       None
;  1345          1871  1 !
;  1346          1872  1 ! IMPLICIT INPUTS:
;  1347          1873  1 !
;  1348          1874  1 !       RECORD_MODE - Indicates whether record mode I/O is required
;  1349          1875  1 !       INFILE_FAB - Input file FAB
;  1350          1876  1 !       INFILE_RAB - Input file RAB
;  1351          1877  1 !
;  1352          1878  1 ! IMPLICIT OUTPUTS:
;  1353          1879  1 !
;  1354          1880  1 !       RECORD_COUNT - Number of input file records copied
;  1355          1881  1 !       BLOCK_COUNT - Number of input file blocks copied
;  1356          1882  1 !
;  1357          1883  1 ! COMPLETION CODES:
;  1358          1884  1 !
;  1359          1885  1 !       OK = successful copy
;  1360          1886  1 !       ERROR = I/O error during copy
;  1361          1887  1 !
;  1362          1888  1 ! SIDE EFFECTS:
;  1363          1889  1 !
;  1364          1890  1 !       None
;  1365          1891  1 !
;  1366          1892  1 !--
;  1367          1893  1
;  1368          1894  2     BEGIN
;  1369          1895  2
;  1370          1896  2     LOCAL
;  1371          1897  2         NEXT_READ;                                       ! Temporary buffer pointer
;  1372          1898  2
;  1373          1899  2 !
;  1374          1900  2 ! Initialization
;  1375          1901  2 !
;  1376          1902  2
;  1377          1903  2     RECORD_COUNT = 0;                                    ! Zero the input file record
;  1378          1904  2     BLOCK_COUNT = 0;                                     ! and block counters.
;  1379          1905  2
;  1380          1906  2 !
;  1381          1907  2 ! If necessary, copy the input file to the output file one record at a time.
;  1382          1908  2 !
;  1383          1909  2
;  1384          1910  2     IF .RECORD_MODE                                      ! Test the record mode I/O indicator.
;  1385          1911  2     THEN
```

```
;  1386     1912  2          WHILE 1 DO                                      ! Beginning of the record copying loop which
;  1387     1913  2              BEGIN                                       ! will be terminated by a RETURN in the event
;  1388     1914  2                                                          ! of an input end-of-file or any I/O error.
;  1389     1915  2
;  1390     1916  2              IF NOT $RMS_GET( RAB = INFILE_RAB )         ! Get one record from the input file.
;  1391     1917  2
;  1392     1918  2              THEN                                        ! If the get was not successful,
;  1393     1919  2                  BEGIN                                   ! begin error processing.
;  1394     1920  2
;  1395     1921  2                  IF .INFILE_RAB[RAB$L_STS] EQL RMS$_EOF  ! If the error was an input end-of-file,
;  1396     1922  2                  THEN
;  1397     1923  2                      RETURN OK;                          ! return a success code to the caller.
;  1398     1924  2
;  1399     1925  2                  IN_READ_ERROR();                        ! Otherwise, send an error message to the user
;  1400     1926  2                  RETURN ERROR;                           ! and return an error code to the caller.
;  1401     1927  2
;  1402     1928  2                  END;                                    ! End of input error processing.
;  1403     1929  2
;  1404     1930  2              OUTFILE_RAB[RAB$L_RBF] =                    ! Copy the input record address
;  1405     1931  2                          .INFILE_RAB[RAB$L_RBF];         ! and record length
;  1406     1932  2              OUTFILE_RAB[RAB$W_RSZ] =                    ! from the input file RAB
;  1407     1933  2                          .INFILE_RAB[RAB$W_RSZ];         ! to the output file RAB.
;  1408     1934  2
;  1409     1935  2              IF $RMS_PUT( RAB = OUTFILE_RAB )            ! Write one record into the output file.
;  1410     1936  2
;  1411     1937  2              THEN                                        ! If the put was successful,
;  1412     1938  2                  RECORD_COUNT = .RECORD_COUNT + 1         ! increment the record counter.
;  1413     1939  2              ELSE
;  1414     1940  2                  BEGIN                                   ! Otherwise,
;  1415     1941  2                  OUT_WRITE_ERROR();                      ! send an error message to the user
;  1416     1942  2                  RETURN ERROR;                           ! and return to the caller.
;  1417     1943  2                  END;
;  1418     1944  2
;  1419     1945  2              END                                         ! End of record mode copy loop.
;  1420     1946  2
;  1421     1947  1
;  1422     1948  1      !
;  1423     1949  1      ! If possible, copy the input file to the output file a block at a time.
;  1424     1950  1      !
;  1425     1951  2          ELSE
;  1426     1952  2              WHILE 1 DO                                  ! Beginning of the block copying loop which
;  1427     1953  2                  BEGIN                                   ! will be terminated by a RETURN in the event
;  1428     1954  2                                                          ! of an input end-of-file or any I/O error.
;  1429     1955  2
;  1430     1956  2                  $RMS_READ( RAB = INFILE_RAB );          ! Begin an asynchronous read from the input file.
;  1431     1957  2
;  1432     1958  2                  IF NOT $RMS_WAIT( RAB = OUTFILE_RAB )   ! Wait for the previous write to complete.
;  1433     1959  2                  THEN
;  1434     1960  2                      BEGIN                               ! If the write was not successful,
;  1435     1961  2                      OUT_WRITE_ERROR();                  ! send the user an error message,
;  1436     1962  2                      $RMS_WAIT( RAB = INFILE_RAB );      ! wait for the previous read to complete,
;  1437     1963  2                      RETURN ERROR;                       ! and then return an error code to the caller.
;  1438     1964  2                      END;
;  1439     1965  2
;  1440     1966  2                  IF $RMS_WAIT( RAB = INFILE_RAB )        ! Wait for the previous read to complete.
;  1441     1967  3                  THEN
;  1442     1968  4                      BEGIN                               ! If the read was successful,
```

```
;  1443          1969  4                INFILE_RAB[RAB$L_UBF] =                     ! save the current output buffer address
;  1444          1970  4                        .OUTFILE_RAB[RAB$L_RBF];            !
;  1445          1971  4                OUTFILE_RAB[RAB$L_RBF] =                    ! and copy the input block address and block size
;  1446          1972  4                        .INFILE_RAB[RAB$L_RBF];             ! from the input file RAB into the output RAB.
;  1447          1973  4                OUTFILE_RAB[RAB$W_RSZ] =                    !
;  1448          1974  4                        .INFILE_RAB[RAB$W_RSZ];             !
;  1449          1975  4                                                            !
;  1450          1976  4                $RMS_WRITE( RAB = OUTFILE_RAB );            ! Initiate an asynchronous write.
;  1451          1977  4                                                            !
;  1452          1978  4                BLOCK_COUNT = .BLOCK_COUNT +                ! Increment the count of blocks written.
;  1453          1979  5                    (.INFILE_RAB[RAB$W_RSZ] +               !
;  1454          1980  4                    .BLOCK_SIZE - 1) / .BLOCK_SIZE;         !
;  1455          1981  4                END                                         !
;  1456          1982  3            ELSE                                            ! If the read was unsuccessful,
;  1457          1983  4                BEGIN                                       ! begin special input error processing.
;  1458          1984  4                                                            !
;  1459          1985  4                IF .INFILE_RAB[RAB$L_STS] EQL RMS$_EOF      ! If the error was an input end-of-file,
;  1460          1986  4                THEN                                        !
;  1461          1987  4                    RETURN OK;                              ! return a success code to the caller.
;  1462          1988  4                                                            !
;  1463          1989  4                IN_READ_ERROR();                            ! Otherwise, send an error message to the user
;  1464          1990  4                RETURN ERROR;                               ! and then return an error code to the caller.
;  1465          1991  3                END;                                        !
;  1466          1992  3                                                            !
;  1467          1993  2            END;                                            ! End of block mode copy loop.
;  1468          1994  2                                                            !
;  1469          1995  2        RETURN OK;
;  1470          1996  1        END;


                                                .EXTRN  SYS$GET, SYS$PUT
                                                .EXTRN  SYS$READ, SYS$WAIT
                                                .EXTRN  SYS$WRITE

                                                .PSECT  COPY$COPY_FILE,NOWRT,9

                                    003C 00000 COPY_FILE:
                                                .WORD   Save R2,R3,R4,R5                ;  1857
                    55      0000'   CF   9E 00002       MOVAB   BLOCK_COUNT, R5         ;
                    54  00000000G    00   9E 00007       MOVAB   SYS$WAIT, R4            ;
                    53      0000G   CF   9E 0000E       MOVAB   OUTFILE_RAB+40, R3      ;
                    52      0000G   CF   9E 00013       MOVAB   INFILE_RAB, R2          ;
                                    65   7C 00018       CLRQ    BLOCK_COUNT             ;  1904
2E      36  A5                      06   E1 0001A       BBC     #6, COPY$SEM_STATUS+2, 3$       ;  1910
                                    52   DD 0001F 1$:    PUSHL   R2                      ;  1916
    00000000G   00                  01   FB 00021       CALLS   #1, SYS$GET             ;
                72                  50   E9 00028       BLBC    R0, 5$                  ;
                63      28      A2  D0 0002B           MOVL    INFILE_RAB+40, OUTFILE_RAB+40   ;  1931
          FA    A3      22      A2  B0 0002F           MOVW    INFILE_RAB+34, OUTFILE_RAB+34   ;  1933
                        D8      A3  9F 00034           PUSHAB  OUTFILE_RAB             ;  1935
    00000000G   00                  01   FB 00037       CALLS   #1, SYS$PUT             ;
                05                  50   E9 0003E       BLBC    R0, 2$                  ;
                            04  A5  D6 00041           INCL    RECORD_COUNT            ;  1938
                                    D9   11 00044       BRB     1$                      ;
          0000V CF                  00   FB 00046 2$:    CALLS   #0, OUT_WRITE_ERROR     ;  1941
                                    5F   11 0004B       BRB     6$                      ;  1942
```

```
                                                 52  DD 0004D 3$:    PUSHL   R2                                      ; 1956
                                                 01  FB 0004F       CALLS   #1, SYS$READ                            ;
                                      00000000G  00                                                                  ;
                                              DB  A3  9F 00056       PUSHAB  OUTFILE_RAB                             ; 1958
                                                 01  FB 00059       CALLS   #1, SYS$WAIT                            ;
                                                 64                                                                 ;
                                                 0C  50  E8 0005C       BLBS    R0, 4$                              ;
                                           0000V CF  00  FB 0005F       CALLS   #0, OUT_WRITE_ERROR                     ; 1961
                                                 52  DD 00064       PUSHL   R2                                      ; 1962
                                                 64  01  FB 00066       CALLS   #1, SYS$WAIT                            ;
                                                 41  11  00069       BRB     6$                                      ; 1963
                                                 52  DD 0006B 4$:    PUSHL   R2                                      ; 1966
                                                 64  01  FB 0006D       CALLS   #1, SYS$WAIT                            ;
                                                 2A  50  E9 00070       BLBC    R0, 5$                              ;
                                              24 A2  63  D0 00073       MOVL    OUTFILE_RAB+40, INFILE_RAB+36           ; 1970
                                                 63  28  A2 D0 00077       MOVL    INFILE_RAB+40, OUTFILE_RAB+40           ; 1972
                                              FA A3  22  A2 B0 0007B       MOVW    INFILE_RAB+34, OUTFILE_RAB+34           ; 1974
                                                 D8  A3  9F 00080       PUSHAB  OUTFILE_RAB                             ; 1976
                                      00000000G  00  01  FB 00083       CALLS   #1, SYS$WRITE                           ;
                                                 50  22  A2 3C 0008A       MOVZWL  INFILE_RAB+34, R0                       ; 1980
                                                 50  14  A5 C0 0008E       ADDL2   BLOCK_SIZE, R0                          ;
                                                 50  D7  00092       DECL    R0                                      ; 1979
                                                 50  14  A5 C6 00094       DIVL2   BLOCK_SIZE, R0                          ; 1980
                                                 65  50  C0 00098       ADDL2   R0, BLOCK_COUNT                         ;
                                                 B0  11  0009B       BRB     3$                                      ; 1966
                                      0001827A   8F  08  A2 D1 0009D 5$:    CMPL    INFILE_RAB+8, #98938            ; 1985
                                                 09  13  000A5       BEQL    7$                                      ;
                                           0000V CF  00  FB 000A7       CALLS   #0, IN_READ_ERROR                       ; 1989
                                                 50  02  D0 000AC 6$:    MOVL    #2, R0                              ; 1990
                                                     04  000AF       RET                                             ;
                                                 50  01  D0 000B0 7$:    MOVL    #1, R0                              ; 1995
                                                     04  000B3       RET                                             ; 1996

; Routine Size:  180 bytes,    Routine Base:  COPY$COPY_FILE + 0000

; 1471          1997  1 PSECT CODE = $CODE$;                                  ! Resume the default PSECT (see previous routine).
```

```
;  1473     1998  1 ROUTINE CLOSE_INFILE :  NOVALUE =                    ! Close the current input file
;  1474     1999  1
;  1475     2000  1 !++
;  1476     2001  1 ! FUNCTIONAL DESCRIPTION:
;  1477     2002  1 !
;  1478     2003  1 !       This routine closes the current input file.
;  1479     2004  1 !
;  1480     2005  1 ! FORMAL PARAMETERS:
;  1481     2006  1 !
;  1482     2007  1 !       None
;  1483     2008  1 !
;  1484     2009  1 ! IMPLICIT INPUTS:
;  1485     2010  1 !
;  1486     2011  1 !       INFILE_OPEN - Input file open indicator
;  1487     2012  1 !       INFILE_FAB - Input file FAB
;  1488     2013  1 !
;  1489     2014  1 ! IMPLICIT OUTPUTS:
;  1490     2015  1 !
;  1491     2016  1 !       INFILE_OPEN - Set to indicate that the input file is not open
;  1492     2017  1 !       INFILE_FAB - Input file FAB closed
;  1493     2018  1 !
;  1494     2019  1 ! ROUTINE VALUE:
;  1495     2020  1 !
;  1496     2021  1 !       None
;  1497     2022  1 !
;  1498     2023  1 ! SIDE EFFECTS:
;  1499     2024  1 !
;  1500     2025  1 !       None
;  1501     2026  1 !
;  1502     2027  1 !--
;  1503     2028  1
;  1504     2029  2     BEGIN
;  1505     2030  2
;  1506     2031  2 !
;  1507     2032  2 ! Return to the caller if the input file is not open.
;  1508     2033  2 !
;  1509     2034  2
;  1510     2035  2     IF NOT .INFILE_OPEN                                 ! If the input file is not open,
;  1511     2036  2     THEN                                                !
;  1512     2037  2         RETURN;                                         ! return to the caller.
;  1513     2038  2
;  1514     2039  2     INFILE_OPEN = NO;                                   ! Otherwise, turn off the open indicator.
;  1515     2040  2
;  1516     2041  2 !
;  1517     2042  2 ! Close the input file.
;  1518     2043  2 !
;  1519     2044  2
;  1520   P 2045  2     $RMS_CLOSE( FAB = INFILE_FAB,                       ! Close the input file FAB,
;  1521     2046  2                 ERR = IN_CLOSE_ERROR );                 ! specifying an error action routine.
;  1522     2047  2
;  1523     2048  2 !
;  1524     2049  2 ! Return to the caller.
;  1525     2050  2 !
;  1526     2051  2
;  1527     2052  2     RETURN;                                             ! Return to the caller.
;  1528     2053  2
;  1529     2054  1     END;
```

```
                                                        .EXTRN  SYS$CLOSE

                                                        .PSECT  $CODE$,NOWRT,2

                                        0000 00000 CLOSE_INFILE:
                                                        .WORD   Save nothing                            ; 1998
                14      0000'  CF               02  E1 00002          BBC     #2, COPY$SEM_STATUS+2, 1$  ; 2035
                        0000'  CF               04  8A 00008          BICB2   #4, COPY$SEM_STATUS+2      ; 2039
                                        0000V   CF  9F 0000D          PUSHAB  IN_CLOSE_ERROR             ; 2046
                                        0000G   CF  9F 00011          PUSHAB  INFILE_FAB                 ;
            00000000G   00              02  FB 00015          CALLS   #2, SYS$CLOSE                      ;
                                            04 0001C 1$:      RET                                        ; 2054

; Routine Size:  29 bytes,    Routine Base:  $CODE$ + 065B
```

```
;  1531          2055  1 GLOBAL ROUTINE COPY$CLOSE_OUTF :  NOVALUE =              ! Close the current output file
;  1532          2056  1
;  1533          2057  1 !++
;  1534          2058  1 ! FUNCTIONAL DESCRIPTION:
;  1535          2059  1 !
;  1536          2060  1 !       This routine closes the current output file.
;  1537          2061  1 !
;  1538          2062  1 ! FORMAL PARAMETERS:
;  1539          2063  1 !
;  1540          2064  1 !       None
;  1541          2065  1 !
;  1542          2066  1 ! IMPLICIT INPUTS:
;  1543          2067  1 !
;  1544          2068  1 !       OUTFILE_OPEN - Output file open indicator
;  1545          2069  1 !       OUTFILE_FAB - Output file FAB
;  1546          2070  1 !       TRUNCATE_BIT in COPY$CLI_STATUS if /TRUNCATE was specified.
;  1547          2071  1 !
;  1548          2072  1 ! IMPLICIT OUTPUTS:
;  1549          2073  1 !
;  1550          2074  1 !       OUTFILE_OPEN - Set to indicate that the output file is not open
;  1551          2075  1 !       OUTFILE_FAB - Output file FAB closed
;  1552          2076  1 !
;  1553          2077  1 ! ROUTINE VALUE:
;  1554          2078  1 !
;  1555          2079  1 !       None
;  1556          2080  1 !
;  1557          2081  1 ! SIDE EFFECTS:
;  1558          2082  1 !
;  1559          2083  1 !       File is truncated if /TRUNCATE was specified.
;  1560          2084  1 !
;  1561          2085  1 !--
;  1562          2086  1
;  1563          2087  2     BEGIN
;  1564          2088  2
;  1565          2089  2 !
;  1566          2090  2 ! Return to the caller if the output file is not open.
;  1567          2091  2 !
;  1568          2092  2
;  1569          2093  2     IF NOT .OUTFILE_OPEN                                ! If the output file is not open,
;  1570          2094  2     THEN                                                !
;  1571          2095  2         RETURN OK;                                      ! return a success code to the caller.
;  1572          2096  2
;  1573          2097  2     OUTFILE_OPEN = NO;                                  ! Otherwise, turn off the open indicator.
;  1574          2098  2
;  1575          2099  2 !
;  1576          2100  2 ! Close the output file.
;  1577          2101  2 !
;  1578          2102  2
;  1579        P 2103  2     $RMS_CLOSE( FAB = OUTFILE_FAB,                      ! Close the output file FAB,
;  1580          2104  2                 ERR = COPY$OCLOSE_ERR );                ! specifying an error action routine.
;  1581          2105  2
;  1582          2106  2 !
;  1583          2107  2 ! Reset the incompatible messages flag to FALSE for the next output file. This message
;  1584          2108  2 ! indicates whether an incompatible attributes has been output for an output file.
;  1585          2109  2 !
;  1586          2110  2
;  1587          2111  2     COPY$B_INCOMPAT = FALSE;                            ! Reset incompatible flag
```

```
;  1588          2112  2
;  1589          2113  2 !++
;  1590          2114  2 ! Return to the caller.
;  1591          2115  2 !
;  1592          2116  2 !--
;  1593          2117  2     RETURN;                                 ! Return to the caller.
;  1594          2118  2
;  1595          2119  1     END;



                                        0000 00000          .ENTRY  COPY$CLOSE_OUTF, Save nothing                   ; 2055
                 18     0000' CF             01  E1 00002          BBC     #1, COPY$SEM_STATUS+2, 1$               ; 2093
                        0000' CF             02  8A 00008          BICB2   #2, COPY$SEM_STATUS+2                   ; 2097
                                      0000V  CF  9F 0000D          PUSHAB  COPY$CLOSE_ERR                          ; 2104
                                      0000G  CF  9F 00011          PUSHAB  OUTFILE_FAB
                           00000000G  00         02  FB 00015          CALLS   #2, SYS$CLOSE
                                      0000'  CF  94 0001C          CLRB    COPY$B_INCOMPAT                         ; 2111
                                             04 00020 1$:          RET                                             ; 2119

; Routine Size:  33 bytes,    Routine Base:  $CODE$ + 0678
```

```
;  1597          2120  1 ROUTINE BYPASS_CONCAT =                                  ! Bypass concatenated input files
;  1598          2121  1
;  1599          2122  1 !++
;  1600          2123  1 ! FUNCTIONAL DESCRIPTION:
;  1601          2124  1 !
;  1602          2125  1 !       This routine scans past concatenated input file-specifications.
;  1603          2126  1 !
;  1604          2127  1 ! FORMAL PARAMETERS:
;  1605          2128  1 !
;  1606          2129  1 !       None
;  1607          2130  1 !
;  1608          2131  1 ! IMPLICIT INPUTS:
;  1609          2132  1 !
;  1610          2133  1 !       Bits in the status words COPY$CLI_STATUS and COPY$SEM_STATUS:
;  1611          2134  1 !
;  1612          2135  1 !               APPEND_COMMAND  - APPEND command indicator
;  1613          2136  1 !               CONCAT_FOLLOWS  - concatentation is occurring
;  1614          2137  1 !
;  1615          2138  1 !       INFILE_DESC - Input file request descriptor
;  1616          2139  1 !       CLEANUP_DESC - Input file cleanup request descriptor
;  1617          2140  1 !
;  1618          2141  1 ! IMPLICIT OUTPUTS:
;  1619          2142  1 !
;  1620          2143  1 !       CONCAT_FOLLOWS - Concatenation active indicator turned off
;  1621          2144  1 !       WILDCARD_ACTIVE - Wildcard active indicator turned off
;  1622          2145  1 !
;  1623          2146  1 ! ROUTINE VALUE:
;  1624          2147  1 !
;  1625          2148  1 !       None
;  1626          2149  1 !
;  1627          2150  1 ! SIDE EFFECTS:
;  1628          2151  1 !
;  1629          2152  1 !       INFILE_DESC - Input file request descriptor filled in by the CLI
;  1630          2153  1 !       CLEANUP_DESC - Input file cleanup request descriptor filled in by the CLI
;  1631          2154  1 !
;  1632          2155  1 !--
;  1633          2156  1
;  1634          2157  2     BEGIN
;  1635          2158  2
;  1636          2159  2     LOCAL
;  1637          2160  2         DESC : $BBLOCK[ DSC$C_S_BLN ]                  ! Descriptor for input file name
;  1638          2161  2         ;
;  1639          2162  2
;  1640          2163  2 !
;  1641          2164  2 ! Initialize descriptor.
;  1642          2165  2 !
;  1643          2166  2     CH$FILL( 0, DSC$C_S_BLN, DESC);
;  1644          2167  2     DESC[ DSC$B_CLASS ] = DSC$K_CLASS_D;
;  1645          2168  2
;  1646          2169  2 !
;  1647          2170  2 ! Return to the caller if input concatenation is not active.
;  1648          2171  2 !
;  1649          2172  2
;  1650          2173  2     IF NOT .APPEND_COMMAND AND                             ! If this is a COPY command
;  1651          2174  2         NOT .CONCAT_FOLLOWS                            ! and no input concatentation is active,
;  1652          2175  2     THEN                                                   !
;  1653          2176  2         RETURN false                                       !    then return to the caller.
```

```
2      ELSE
2          CONCAT_FOLLOWS = NO;                                   ! Otherwise, turn off the concatenation indicator.
2
2  !
2  ! Report an wildcard specification which has not been completely processed.
2  !
2
2      IF .WILDCARD_ACTIVE                                        ! If a wildcard spec is currently active,
2      THEN                                                       !
3          BEGIN                                                  !
3          WILDCARD_ACTIVE = NO;                                  ! turn off the wildcard indicator.
3
3          IF .INFILE_NAM_BLK[NAM$B_RSL] NEQ 0                    ! If the wildcard spec is partially processed,
3          THEN                                                   !
4              BEGIN                                              !
4              INFILE_NAM_BLK[NAM$B_RSL] = 0;                     ! discard the current resultant name string,
4              REPORT_BYPASS( MSG$_NOTCMPLT );                    ! and report the bypass wildcard spec.
3              END;
2          END;
2
2  !
2  ! Scan past any concatenated input file-specifications.
2  !
2
2      WHILE CLI$GET_VALUE( $DESCRIPTOR('INFILE'), DESC ) DO
2
2          IF COPY$FIND_INPUT_FILE( DESC )                        ! Parse the input file-specification.
2          THEN
2              REPORT_BYPASS( MSG$_NOTCOPIED );                   ! Report that the file was not processed.
2
2  !
2  ! Return to the caller.
2  !
2
2      RETURN true;                                               ! Return to the caller.
2
1      END;


                                                      .PSECT  $PLIT$,NOWRT,NOEXE,2

              45  4C  49  46  4E  49  0004C P.AAH:    .ASCII  \INFILE\                              ;
                                      00052           .BLKB   2
                            00000006  00054 P.AAG:    .LONG   6                                     ;
                            00000000' 00058           .ADDRESS P.AAH                                ;


                                                      .PSECT  $CODE$,NOWRT,2

                                003C  00000 BYPASS_CONCAT:
                                                      .WORD   Save R2,R3,R4,R5                      ; 2120
                      5E          08  C2 00002        SUBL2   #8, SP                                ;
   08         00      6E          00  2C 00005        MOVC5   #0, (SP), #0, #8, DESC                ; 2166
                                  6E     0000A                                                      ;
                 03   AE          02  90 0000B        MOVB    #2, DESC+3                            ; 2167
```

```
                          06      0000'   CF  E8  0000F           BLBS    COPY$CLI_STATUS, 1$                         ;2173
                49        0000'   CF      03  E1  00014           BBC     #3, COPY$SEM_STATUS+2, 5$                   ;2174
                          0000'   CF      08  8A  0001A 1$:       BICB2   #8, COPY$SEM_STATUS+2                       ;2178
                19        0000'   CF      05  E1  0001F           BBC     #5, COPY$SEM_STATUS+2, 3$                   ;2184
                          0000'   CF      20  8A  00025           BICB2   #32, COPY$SEM_STATUS+2                      ;2187
                                  0000G   CF  95  0002A           TSTB    INFILE_NAM_BLK+3                            ;2189
                                          0E  13  0002E           BEQL    3$                                          ;
                                  0000G   CF  94  00030           CLRB    INFILE_NAM_BLK+3                            ;2192
                          7E      11C0    8F  3C  00034           MOVZWL  #4544, -(SP)                                ;2193
                    0000V CF              01  FB  00039 2$:       CALLS   #1, REPORT_BYPASS                           ;
                                          5E  DD  0003E 3$:       PUSHL   SP                                          ;2201
                                  0000'   CF  9F  00040           PUSHAB  P.AAG                                       ;
                00000000G 00              02  FB  00044           CALLS   #2, CLI$GET_VALUE                           ;
                          11              50  E9  0004B           BLBC    R0, 4$                                      ;
                                          5E  DD  0004E           PUSHL   SP                                          ;2203
                    0000V CF              01  FB  00050           CALLS   #1, COPY$FIND_INPUT_FILE                    ;
                          E6              50  E9  00055           BLBC    R0, 3$                                      ;
                          7E      11B8    8F  3C  00058           MOVZWL  #4536, -(SP)                                ;2205
                                          DA  11  0005D           BRB     2$                                          ;
                          50              01  D0  0005F 4$:       MOVL    #1, R0                                      ;2211
                                              04  00062           RET                                                 ;
                                          50  D4  00063 5$:       CLRL    R0                                          ;2213
                                              04  00065           RET                                                 ;

; Routine Size:  102 bytes,    Routine Base:  $CODE$ + 0699
```

```
;  1692          2214  1 GLOBAL ROUTINE COPY$FIND_INPUT_FILE ( INFILE_DESC : REF $BBLOCK ) =
;  1693          2215  1
;  1694          2216  1 !++
;  1695          2217  1 ! FUNCTIONAL DESCRIPTION:
;  1696          2218  1 !
;  1697          2219  1 !       This routine calls RMS to parse an input file-specification.
;  1698          2220  1 !
;  1699          2221  1 ! FORMAL PARAMETERS:
;  1700          2222  1 !
;  1701          2223  1 !       None
;  1702          2224  1 !
;  1703          2225  1 ! IMPLICIT INPUTS:
;  1704          2226  1 !
;  1705          2227  1 !       INFILE_FAB - Input file FAB
;  1706          2228  1 !       INFILE_NAM_BLK - Input file name block
;  1707          2229  1 !
;  1708          2230  1 ! IMPLICIT OUTPUTS:
;  1709          2231  1 !
;  1710          2232  1 !       INFILE_FAB - FNA and FNS fields filled in.
;  1711          2233  1 !
;  1712          2234  1 ! COMPLETION CODES:
;  1713          2235  1 !
;  1714          2236  1 !       OK = Successful parse
;  1715          2237  1 !       ERROR = Error from RMS parse
;  1716          2238  1 !
;  1717          2239  1 ! SIDE EFFECTS:
;  1718          2240  1 !
;  1719          2241  1 !       None
;  1720          2242  1 !
;  1721          2243  1 !--
;  1722          2244  1
;  1723          2245  2     BEGIN
;  1724          2246  2
;  1725          2247  2     OWN
;  1726          2248  2         find_file_context : INITIAL(0);                 ! Context parameter for LIB$FIND_FILE
;  1727          2249  2
;  1728          2250  2     LOCAL
;  1729          2251  2         resultant_name_desc : $BBLOCK[ DSC$C_S_BLN ],   ! Descriptor for filespec returned by LIB$FIND_FILE
;  1730          2252  2         find_file_nam : REF $BBLOCK[],                  ! Pointer to NAM block used by LIB$FIND_FILE
;  1731          2253  2         status;                                         ! Status returned by LIB$FIIND_FILE
;  1732          2254  2
;  1733          2255  2     BIND
;  1734          2256  2         find_file_fab = find_file_context : REF $BBLOCK[];
;  1735          2257  2
;  1736          2258  2
;  1737          2259  2
;  1738          2260  2     ! Initialize the descriptor for the resultant name string.
;  1739          2261  2     !
;  1740          2262  2     CH$FILL( 0, DSC$C_S_BLN, resultant_name_desc );
;  1741          2263  2     resultant_name_desc[ DSC$B_CLASS ] = DSC$K_CLASS_D;
;  1742          2264  2
;  1743          2265  2
;  1744          2266  2     ! Zero the expanded name sting length, so that COPY$INOPN_ERR can determine
;  1745          2267  2     ! if the expanded string was created by RMS or not.
;  1746          2268  2     !
;  1747          2269  2     INFILE_NAM_BLK[NAM$B_ESL] = 0;
;  1748          2270  2
```

```
;  1749          2271  2
;  1750          2272  2      ! Call LIB$FIND_FILE to locate the file.  If something other than success is
;  1751          2273  2      ! returned, then check to see if it is something we care about.  NMF, no
;  1752          2274  2      ! more files doesn't matter,  for any other error condition COPY should
;  1753          2275  2      ! issue a message.
;  1754          2276  2      !
;  1755          2277  2      IF NOT ( status = LIB$FIND_FILE( .infile_desc, resultant_name_desc,
;  1756          2278  2              find_file_context, 0, 0, 0, %ref(2)))
;  1757          2279  2      THEN
;  1758          2280  2          BEGIN
;  1759          2281  2          IF .status NEQ RMS$_NMF
;  1760          2282  2          THEN
;  1761          2283  2              COPY$INOPN_ERR( .find_file_context );
;  1762          2284  2          RETURN .status;
;  1763          2285  2          END;
;  1764          2286  2
;  1765          2287  2
;  1766          2288  2      ! Copy the information from the resultant name string descripitor into
;  1767          2289  2      ! the FAB's file name and the NAM block's resultant name descriptor fields.
;  1768          2290  2      ! Also, copy the file name status bits into the input file's NAM block and
;  1769          2291  2      ! copy the FID of the found file into the input file's name block. (COPY
;  1770          2292  2      ! does an open by name block. This guarantees that the correct file is
;  1771          2293  2      ! opened.).  Then return to the caller.
;  1772          2294  2      !
;  1773          2295  2      infile_fab[ FAB$L_FNA ]     = .resultant_name_desc[ DSC$A_POINTER ];
;  1774          2296  2      infile_fab[ FAB$B_FNS ]     = .resultant_name_desc[ DSC$W_LENGTH ];
;  1775          2297  2      infile_nam_blk[ NAM$B_RSL ] = .resultant_name_desc[ DSC$W_LENGTH ];
;  1776          2298  2      in_name_desc[ 0 ]           = .infile_nam_blk[ NAM$B_RSL ];
;  1777          2299  2      CH$MOVE(.infile_fab[FAB$B_FNS], .infile_fab[FAB$L_FNA], .in_name_desc[1]);
;  1778          2300  2
;  1779          2301  2
;  1780          2302  2      find_file_nam = .find_file_fab[ FAB$L_NAM ];
;  1781          2303  2      infile_nam_blk[ NAM$L_FNB ]     = .find_file_nam[ NAM$L_FNB ];
;  1782          2304  2      infile_nam_blk[ NAM$W_FID_NUM ] = .find_file_nam[ NAM$W_FID_NUM ];
;  1783          2305  2      infile_nam_blk[ NAM$W_FID_SEQ ] = .find_file_nam[ NAM$W_FID_SEQ ];
;  1784          2306  2      infile_nam_blk[ NAM$W_FID_RVN ] = .find_file_nam[ NAM$W_FID_RVN ];
;  1785          2307  2      CH$MOVE( NAM$S_DVI, find_file_nam[NAM$T_DVI], infile_nam_blk[NAM$T_DVI] );
;  1786          2308  2
;  1787          2309  2      RETURN ok;
;  1788          2310  2
;  1789          2311  1      END;


                                                .PSECT  $OWN$,NOEXE,2

                                00000000  00000 FIND_FILE_CONTEXT:
                                                .LONG   0                                                       ;

                                                FIND_FILE_FAB=      FIND_FILE_CONTEXT


                                                .PSECT  $CODE$,NOWRT,2

                                    00FC 00000          .ENTRY  COPY$FIND_INPUT_FILE, Save R2,R3,R4,R5,R6,- ; 2214
                                                                R7                                          ;
                              57    0000'  CF  9E 00002          MOVAB   FIND_FILE_CONTEXT, R7                   ;
```

```
                                56      0000G   CF   9E 00007         MOVAB   INFILE_NAM_BLK+3, R6                                  ;
                                5E              0C   C2 0000C         SUBL2   #12, SP                                             ;
        08              00      6E              00   2C 0000F         MOVC5   #0, (SP), #0, #8, RESULTANT_NAME_DESC               ; 2262
                                            04  AE   00014
                            07  AE              02   90 00016         MOVB    #2, RESULTANT_NAME_DESC+3                           ; 2263
                                            08  A6   94 0001A         CLRB    INFILE_NAM_BLK+11                                   ; 2269
                                6E              02   D0 0001D         MOVL    #2, (SP)                                            ; 2278
                                                5E   DD 00020         PUSHL   SP
                                                7E   7C 00022         CLRQ    -(SP)                                               ; 2277
                                                7E   D4 00024         CLRL    -(SP)
                                                57   DD 00026         PUSHL   R7
                                            18  AE   9F 00028         PUSHAB  RESULTANT_NAME_DESC
                                            04  AC   DD 0002B         PUSHL   INFILE_DESC
                    00000000G   00              07   FB 0002E         CALLS   #7, LIB$FIND_FILE
                                52              50   D0 00035         MOVL    R0, STATUS
                                14              52   E8 00038         BLBS    STATUS, 2$
                    000182CA    8F              52   D1 0003B         CMPL    STATUS, #99018                                      ; 2281
                                                07   13 00042         BEQL    1$
                                                67   DD 00044         PUSHL   FIND_FILE_CONTEXT                                   ; 2283
                            0000V   CF          01   FB 00046         CALLS   #1, COPY$INOPN_ERR
                                50              52   D0 0004B 1$:     MOVL    STATUS, R0                                          ; 2284
                                                04   0004E           RET
                            0000G   CF      08  AE   D0 0004F 2$:     MOVL    RESULTANT_NAME_DESC+4, INFILE_FAB+44                ; 2295
                            0000G   CF      04  AE   90 00055         MOVB    RESULTANT_NAME_DESC, INFILE_FAB+52                  ; 2296
                                66          04  AE   90 0005B         MOVB    RESULTANT_NAME_DESC, INFILE_NAM_BLK+3               ; 2297
                            0000G   CF          66   9A 0005F         MOVZBL  INFILE_NAM_BLK+3, IN_NAME_DESC                      ; 2298
                                50      0000G   CF   9A 00064         MOVZBL  INFILE_FAB+52, R0                                   ; 2299
        0000G   DF          0000G   DF          50   28 00069         MOVC3   R0, @INFILE_FAB+44, @IN_NAME_DESC+4
                                50              67   D0 00071         MOVL    FIND_FILE_FAB, R0                                   ; 2302
                                50          28  A0   D0 00074         MOVL    40(R0), FIND_FILE_NAM
                            31  A6          34  A0   D0 00078         MOVL    52(FIND_FILE_NAM), INFILE_NAM_BLK+52                ; 2303
                            21  A6          24  A0   D0 0007D         MOVL    36(FIND_FILE_NAM), INFILE_NAM_BLK+36                ; 2304
                            25  A6          28  A0   B0 00082         MOVW    40(FIND_FILE_NAM), INFILE_NAM_BLK+40                ; 2306
                    11  A6      14  A0              10   28 00087     MOVC3   #16, 20(FIND_FILE_NAM), INFILE_NAM_BLK+20           ; 2307
                                50                  01   D0 0008D     MOVL    #1, R0                                              ; 2309
                                                        04   00090     RET                                                         ; 2311

; Routine Size:  145 bytes,    Routine Base:  $CODE$ + 06FF
```

```
;  1791          2312  1 GLOBAL ROUTINE COPY$CALC_ALQ =                        ! Allocation quantity calculation routine
;  1792          2313  1
;  1793          2314  1 !++
;  1794          2315  1 ! FUNCTIONAL DESCRIPTION:
;  1795          2316  1 !
;  1796          2317  1 !       This routine determines the output file allocation/extension quantity.
;  1797          2318  1 !
;  1798          2319  1 ! FORMAL PARAMETERS:
;  1799          2320  1 !
;  1800          2321  1 !       None
;  1801          2322  1 !
;  1802          2323  1 ! IMPLICIT INPUTS:
;  1803          2324  1 !
;  1804          2325  1 !       EXTEND_OUTFILE  - Output file extension indicator
;  1805          2326  1 !       INFILE_FAB      - Input file FAB
;  1806          2327  1 !       INFILE_XABALL   - Input file allocation XAB
;  1807          2328  1 !       INFILE_XABFHC   - Input file header characteristics XAB
;  1808          2329  1 !       COPY$CLI_STATUS bit TRUNCATE_BIT
;  1809          2330  1 !               means /TRUNCATE was specified
;  1810          2331  1 !       ALLOC_VALUE     - contains a value if /ALLOCATION was specified.
;  1811          2332  1 !       COPY_TRUN_QUAL  - CLI data block for the truncate qualifier; the
;  1812          2333  1 !                         "explicit bit" will be set if /NOTRUNCATE was
;  1813          2334  1 !                         specified on the input line
;  1814          2335  1 !
;  1815          2336  1 ! IMPLICIT OUTPUTS:
;  1816          2337  1 !
;  1817          2338  1 !       None
;  1818          2339  1 !
;  1819          2340  1 ! ROUTINE VALUE:
;  1820          2341  1 !
;  1821          2342  1 !       Size of the input file (i.e., number of blocks)
;  1822          2343  1 !
;  1823          2344  1 ! SIDE EFFECTS:
;  1824          2345  1 !
;  1825          2346  1 !       None
;  1826          2347  1 !
;  1827          2348  1 !--
;  1828          2349  1
;  1829          2350  2     BEGIN
;  1830          2351  2
;  1831          2352  2     LOCAL
;  1832          2353  2         ALQ;                                          ! Temporary allocation quantity
;  1833          2354  2
;  1834          2355  2 !
;  1835          2356  2 ! Return a zero allocation size if the output file is not a disk and it is being extended.
;  1836          2357  2 !
;  1837          2358  2
;  1838          2359  2     IF .EXTEND_OUTFILE AND                            ! If the output file is being extended
;  1839          2360  3        (NOT .OUTFILE_FAB[$FAB_DEV(FOD)] OR            ! and it is not a file structured device
;  1840          2361  3         .OUTFILE_FAB[$FAB_DEV(SQD)])                  ! or it is a magnetic tape,
;  1841          2362  2     THEN                                              !
;  1842          2363  2         RETURN 0;                                     ! return a zero allocation size to the caller.
;  1843          2364  2
;  1844          2365  2 !
;  1845          2366  2 ! Determine the output file allocation size from the size and organization of the input file.
;  1846          2367  2 !
;  1847          2368  2
```

```
;  1848          2369  2      IF NOT .INFILE_FAB[$FAB_DEV(FOD)] OR                    ! If the input device is not file structured
;  1849          2370  2          .INFILE_FAB[$FAB_DEV(SQD)]                          ! or if it is a magnetic tape,
;  1850          2371  2      THEN                                                    !
;  1851          2372  2          ALQ = DEFAULT_ALLOC                                 ! assume a default input file size.
;  1852          2373  2      ELSE
;  1853          2374  2          BEGIN
;  1854          2375  3
;  1855          2376  3      !   If the input file is a non-contiguous sequential file and /NOTRUNCATE was not explicitly given
;  1856          2377  3      !       or
;  1857          2378  3      !   the the input file is being appened to an existing file,
;  1858          2379  3      !       or
;  1859          2380  3      !   if /TRUNCATE and no /ALLOCATION was given,
;  1860          2381  3      !   the file should be truncated.  Otherwise, use the allocation of the input file as the size of
;  1861          2382  3      !   the output file.
;  1862          2383  3      !
;  1863          2384  4          IF  (
;  1864          2385  5          (
;  1865          2386  5          .INFILE_FAB[ FAB$B_ORG ] EQL FAB$C_SEQ
;  1866          2387  5              AND NOT
;  1867          2388  6          ( .TRUNCATE_NEGATED OR .NEG_TRUNCATE_QUAL )
;  1868          2389  5          )
;  1869          2390  4              AND
;  1870          2391  5          ( NOT .INFILE_XABALL[ XAB$V_CTG ] OR .EXTEND_OUTFILE )
;  1871          2392  5          )
;  1872          2393  4              OR
;  1873          2394  4          ( (.TRUNCATE_QUAL OR .LOC_TRUNCATE_QUAL) AND .CURR_ALLOCATION_VALUE EQL 0 )
;  1874          2395  3          THEN                                                !
;  1875          2396  3              IF .INFILE_XABFHC[XAB$W_FFB] EQL 0              ! calculate only enough space to hold the actual
;  1876          2397  3              THEN                                            ! data in the input file.  Note that this calculatio
;  1877          2398  4                  ALQ = .INFILE_XABFHC[XAB$L_EBK] - 1         ! includes the final block only if it actually
;  1878          2399  3              ELSE                                            ! contains some data.
;  1879          2400  4                  ALQ = .INFILE_XABFHC[XAB$L_EBK]             !
;  1880          2401  3          ELSE
;  1881          2402  3              ALQ = .INFILE_XABFHC[XAB$L_HBK];                ! Otherwise, pickup the actual size of the input fil
;  1882          2403  3          END;
;  1883          2404  2
;  1884          2405  2      IF .EXTEND_OUTFILE                                      ! If the output file is being extended,
;  1885          2406  2      THEN                                                    !
;  1886          2407  2          ALQ = .OUTFILE_XABFHC[XAB$L_EBK] + .ALQ -           ! subtract the remaining output file space
;  1887          2408  2                .OUTFILE_XABFHC[XAB$L_HBK];                ! from the calculated extension quantity.
;  1888          2409  2
;  1889          2410  2      !
;  1890          2411  2      ! Return the calculated allocation (or extension) quantity to the caller.
;  1891          2412  2      !
;  1892          2413  2
;  1893          2414  2      IF .ALQ GEQ 0                                           ! If the calculated allocation/extension quantity
;  1894          2415  2      THEN                                                    ! is greater than or equal to zero,
;  1895          2416  2          RETURN .ALQ                                         ! return that value to the caller.
;  1896          2417  2      ELSE
;  1897          2418  2          RETURN 0;                                           ! Otherwise, return a zero value to the caller.
;  1898          2419  2
;  1899          2420  1      END;
```

```
                                                   0004 00000       .ENTRY  COPY$CALC_ALQ, Save R2                     ; 2312
                                 52      0000G  CF   9E 00002       MOVAB   INFILE_XABFHC+16, R2                       ;
        51     0000'  CF         01             07   EF 00007       EXTZV   #7, #1, COPY$SEM_STATUS+2, R1              ; 2359
                                 0C             51   E9 0000E       BLBC    R1, 1$                                     ;
                      67  0000G  CF             06   E1 00011       BBC     #6, OUTFILE_FAB+65, 11$                    ; 2360
                      61  0000G  CF             05   E0 00017       BBS     #5, OUTFILE_FAB+64, 11$                    ; 2361
                      06  0000G  CF             06   E1 0001D 1$:   BBC     #6, INFILE_FAB+65, 2$                      ; 2369
                      04  0000G  CF             05   E1 00023       BBC     #5, INFILE_FAB+64, 3$                      ; 2370
                                                50   D4 00029 2$:   CLRL    ALQ                                        ; 2372
                                                40   11 0002B       BRB     9$                                         ;
                                         0000G  CF   95 0002D 3$:   TSTB    INFILE_FAB+29                              ; 2386
                                                14   12 00031       BNEQ    4$                                         ;
                      0E  0000'  CF             06   E0 00033       BBS     #6, COPY$CLI_STATUS+5, 4$                  ; 2388
                                 09      0000'  CF   E8 00039       BLBS    COPY$CLI_STATUS+6, 4$                      ;
                                         0000G  CF   95 0003E       TSTB    INFILE_XABALL+8                            ; 2391
                                                15   18 00042       BGEQ    6$                                         ;
                                 12             51   E8 00044       BLBS    R1, 6$                                     ;
                      06  0000'  CF             05   E0 00047 4$:   BBS     #5, COPY$CLI_STATUS+5, 5$                  ; 2394
                                         0000'  CF   95 0004D       TSTB    COPY$CLI_STATUS+5                          ;
                                                16   18 00051       BGEQ    8$                                         ;
                                         0000G  CF   D5 00053 5$:   TSTL    CURR_ALLOCATION_VALUE                      ;
                                                10   12 00057       BNEQ    8$                                         ;
                                            04  A2   B5 00059 6$:   TSTW    INFILE_XABFHC+20                           ; 2396
                                                06   12 0005C       BNEQ    7$                                         ;
                      50         62             01   C3 0005E       SUBL3   #1, INFILE_XABFHC+16, ALQ                  ; 2398
                                                09   11 00062       BRB     9$                                         ;
                                 50             62   D0 00064 7$:   MOVL    INFILE_XABFHC+16, ALQ                      ; 2400
                                                04   11 00067       BRB     9$                                         ; 2396
                                 50         FC  A2   D0 00069 8$:   MOVL    INFILE_XABFHC+12, ALQ                      ; 2402
                                 0C             51   E9 0006D 9$:   BLBC    R1, 10$                                    ; 2405
                      51         50      0000G  CF   C1 00070       ADDL3   OUTFILE_XABFHC+16, ALQ, R1                 ; 2407
                      50         51      0000G  CF   C3 00076       SUBL3   OUTFILE_XABFHC+12, R1, ALQ                 ; 2408
                                                02   18 0007C 10$:  BGEQ    12$                                        ; 2414
                                                50   D4 0007E 11$:  CLRL    R0                                         ; 2420
                                                     04 00080 12$:  RET                                                ;

; Routine Size:  129 bytes,    Routine Base:  $CODE$ + 0790
```

```
;  1901     2421  1 ROUTINE REPORT_NAMES                                  ! Report the results of a file copy
;  1902     2422  1                     : NOVALUE =
;  1903     2423  1
;  1904     2424  1 !++
;  1905     2425  1 ! FUNCTIONAL DESCRIPTION:
;  1906     2426  1 !
;  1907     2427  1 !       This routine reports the results of copying a single input file
;  1908     2428  1 !       to the output file.
;  1909     2429  1 !
;  1910     2430  1 ! FORMAL PARAMETERS:
;  1911     2431  1 !
;  1912     2432  1 !       None
;  1913     2433  1 !
;  1914     2434  1 ! IMPLICIT INPUTS:
;  1915     2435  1 !
;  1916     2436  1 !       LOG - Indicator tested to see if activity reporting desired
;  1917     2437  1 !       EXTEND_OUTFILE - Indicator tested to see if input concatenation is active.
;  1918     2438  1 !       IN_NAME_DESC - Input file name descriptor
;  1919     2439  1 !       OUT_NAME_DESC - Output file name descriptor
;  1920     2440  1 !       BLOCK_COUNT - Number of input file blocks copied
;  1921     2441  1 !       RECORD_COUNT - Number of input file records copied
;  1922     2442  1 !       INFILE_FAB - Address of input file FAB
;  1923     2443  1 !
;  1924     2444  1 ! IMPLICIT OUTPUTS:
;  1925     2445  1 !
;  1926     2446  1 !       None
;  1927     2447  1 !
;  1928     2448  1 ! ROUTINE VALUE:
;  1929     2449  1 !
;  1930     2450  1 !       None
;  1931     2451  1 !
;  1932     2452  1 ! SIDE EFFECTS:
;  1933     2453  1 !
;  1934     2454  1 !       None
;  1935     2455  1 !
;  1936     2456  1 !--
;  1937     2457  1
;  1938     2458  2     BEGIN
;  1939     2459  2
;  1940     2460  2     LOCAL
;  1941     2461  2         ptr,                                            ! Temporary variables for character searching
;  1942     2462  2         address,
;  1943     2463  2         size;
;  1944     2464  2
;  1945     2465  2 !
;  1946     2466  2 ! Determine which message, if any, is needed.
;  1947     2467  2 !
;  1948     2468  2
;  1949     2469  2     IF NOT .LOG_MSG_QUAL                                ! If activity reporting is not requested,
;  1950     2470  2     THEN                                                !
;  1951     2471  2         RETURN;                                         ! return to the caller.
;  1952     2472  2
;  1953     2473  2 !
;  1954     2474  2 ! If this is a record oriented device (not network), the messages should
;  1955     2475  2 ! include only the device name.
;  1956     2476  2 !
;  1957     2477  2
```

```
;  1958     2478  2
;  1959     2479  2      IF .infile_fab [$FAB_DEV(rec)]
;  1960     2480  2          AND NOT .infile_fab [$FAB_DEV(net)]
;  1961     2481  2      THEN
;  1962     2482  2          BEGIN
;  1963     2483  2          size = .in_name_desc[0];
;  1964     2484  2          address = .in_name_desc[1];
;  1965     2485  2          ptr = CH$FIND_CH(.size,.address,':');
;  1966     2486  2          IF .ptr NEQ 0           ! If there is anything past the device, remove it
;  1967     2487  2          THEN
;  1968     2488  2              in_name_desc[0] =  .ptr - .address + 1;
;  1969     2489  2          END;
;  1970     2490  2
;  1971     2491  2
;  1972     2492  2      IF NOT .EXTEND_OUTFILE                                  ! Test the record mode indicator to see
;  1973     2493  2                                                              ! if this is the primary input file or a
;  1974     2494  2                                                              ! concatenated input file.
;  1975     2495  2  !
;  1976     2496  2  ! Create a "copied" message if the input file just copied was
;  1977     2497  2  ! the first file copied into the output file.
;  1978     2498  2  !
;  1979     2499  2
;  1980     2500  2      THEN                                                    !
;  1981     2501  2          IF .BLOCK_COUNT NEQ 0                               ! If the input file was copied in block mode,
;  1982     2502  2          THEN                                                !
;  1983   P 2503  2              PUT_MESSAGE( MSG$_COPIEDB,                      ! signal "file copied" with the following arguments:
;  1984   P 2504  2                      3,                                      !    Number of message arguments
;  1985   P 2505  2                      IN_NAME_DESC,                           !    Address of input file name descriptor
;  1986   P 2506  2                      OUT_NAME_DESC,                          !    Address of output file name descriptor
;  1987     2507  2                      .BLOCK_COUNT )                          !    Number of blocks copied
;  1988     2508  2
;  1989     2509  2          ELSE                                                ! Otherwise,
;  1990     2510  3              IF (.RECORD_COUNT NEQ 0) OR NOT (LIB$CHECK_DIR (INFILE_FAB)) ! If the input file is not 0 record
;  1991     2511  3                                                              !   is not a directory file
;  1992     2512  2              THEN
;  1993   P 2513  2                  PUT_MESSAGE( MSG$_COPIEDR,                  ! signal "file copied" with the following arguments:
;  1994   P 2514  2                          3,                                  !    Number of message arguments
;  1995   P 2515  2                          IN_NAME_DESC,                       !    Address of input file name descriptor
;  1996   P 2516  2                          OUT_NAME_DESC,                      !    Address of output file name descriptor
;  1997     2517  2                          .RECORD_COUNT )                     !    Number of records copied
;  1998     2518  2
;  1999     2519  2              ELSE                                            ! Otherwise, its a directory file
;  2000   P 2520  2                  PUT_MESSAGE( MSG$_CREATED,                  ! signal "created" with the following arguments:
;  2001   P 2521  2                          1,                                  !    number of message arguments
;  2002     2522  2                          OUT_NAME_DESC )                     !    address of output file descriptor
;  2003     2523  2  !
;  2004     2524  2  ! Create an "appended" message if the input file just copied was
;  2005     2525  2  ! appended to an existing output file.
;  2006     2526  2  !
;  2007     2527  2
;  2008     2528  2      ELSE                                                    !
;  2009     2529  2          IF .BLOCK_COUNT NEQ 0                               ! If the input file was copied in block mode,
;  2010     2530  2          THEN                                                !
;  2011   P 2531  2              PUT_MESSAGE( MSG$_APPENDEDB,                    ! signal "file appended" with the following argument
;  2012   P 2532  2                      3,                                      !    Number of message arguments
;  2013   P 2533  2                      IN_NAME_DESC,                           !    Address of input file name descriptor
;  2014   P 2534  2                      OUT_NAME_DESC,                          !    Address of output file name descriptor
```

```
;  2015          2535  2                        .BLOCK_COUNT )                          !    Number of blocks copied
;  2016          2536  2
;  2017          2537  2                ELSE                                            ! Otherwise,
;  2018        P 2538  2                    PUT_MESSAGE( MSG$_APPENDEDR,                ! signal "file appended" with the following argument
;  2019        P 2539  2                                 3,                             !    Number of message arguments
;  2020        P 2540  2                                 IN_NAME_DESC,                  !    Address of input file name descriptor
;  2021        P 2541  2                                 OUT_NAME_DESC,                 !    Address of output file name descriptor
;  2022          2542  2                                 .RECORD_COUNT );               !    Number of records copied
;  2023          2543  2
;  2024          2544  2    !
;  2025          2545  2    ! Return to the caller.
;  2026          2546  2    !
;  2027          2547  2
;  2028          2548  2        RETURN;                                                 ! Return to the caller.
;  2029          2549  2
;  2030          2550  1        END;


                                        007C 00000 REPORT_NAMES:
                                                           .WORD   Save R2,R3,R4,R5,R6                                     ; 2421
                              56 00000000G   00   9E 00002         MOVAB   LIB$SIGNAL, R6                                    ;
                              55     0000G   CF   9E 00009         MOVAB   OUT_NAME_DESC, R5                                 ;
                              54     0000'   CF   9E 0000E         MOVAB   RECORD_COUNT, R4                                  ;
                              53     0000G   CF   9E 00013         MOVAB   IN_NAME_DESC, R3                                  ;
                    01        14   A4             01   E0 00018         BBS     #1, COPY$CLI_STATUS, 1$                   ; 2469
                                                  04 0001D         RET                                                       ;
                              20     0000G   CF   E9 0001E 1$:      BLBC    INFILE_FAB+64, 3$                         ; 2479
                    1A    0000G   CF             05   E0 00023         BBS     #5, INFILE_FAB+65, 3$                     ; 2480
                                  50             63   D0 00029         MOVL    IN_NAME_DESC, SIZE                        ; 2483
                                  52       04    A3   D0 0002C         MOVL    IN_NAME_DESC+4, ADDRESS                   ; 2484
                    62            50             3A   3A 00030         LOCC    #58, SIZE, (ADDRESS)                      ; 2485
                                                 02   12 00034         BNEQ    2$                                        ;
                                                 51   D4 00036         CLRL    R1                                        ;
                                                 51   D5 00038 2$:      TSTL    PTR                                       ; 2486
                                                 07   13 0003A         BEQL    3$                                        ;
                                  51             52   C2 0003C         SUBL2   ADDRESS, R1                               ; 2488
                                  63       01    A1   9E 0003F         MOVAB   1(R1), IN_NAME_DESC                       ;
                                  50       FC    A4   D0 00043 3$:      MOVL    BLOCK_COUNT, R0                           ; 2501
                                           32    A4   95 00047         TSTB    COPY$SEM_STATUS+2                         ; 2492
                                                 44   19 0004A         BLSS    7$                                        ;
                                                 50   D5 0004C         TSTL    R0                                        ; 2501
                                                 0D   13 0004E         BEQL    4$                                        ;
                                                 50   DD 00050         PUSHL   R0                                        ; 2507
                                                 28   BB 00052         PUSHR   #^M<R3,R5>                                ;
                                                 03   DD 00054         PUSHL   #3                                        ;
                              7E       1061    8F   3C 00056         MOVZWL  #4193, -(SP)                              ;
                                                 4F   11 0005B         BRB     9$                                        ;
                                                 64   D5 0005D 4$:      TSTL    RECORD_COUNT                              ; 2510
                                                 0E   12 0005F         BNEQ    5$                                        ;
                                        0000G    CF   9F 00061         PUSHAB  INFILE_FAB                                ;
                    00000000G   00               01   FB 00065         CALLS   #1, LIB$CHECK_DIR                         ;
                                0D               50   E8 0006C         BLBS    R0, 6$                                    ;
                                                 64   DD 0006F 5$:      PUSHL   RECORD_COUNT                              ; 2517
                                                 28   BB 00071         PUSHR   #^M<R3,R5>                                ;
```

```
                                        03   DD 00073          PUSHL   #3                                          ;
                                7E      1069 8F   3C 00075          MOVZWL  #4201, -(SP)                            ;
                                             30   11 00077          BRB     9$                                      ;
                                             55   DD 0007C 6$:      PUSHL   R5                                      ; 2522
                                             01   DD 0007E          PUSHL   #1                                      ;
                                7E      1073 8F   3C 00080          MOVZWL  #4211, -(SP)                            ;
                        0000V   CF           01   FB 00085          CALLS   #1, COPY$MSG_NUMBER                     ;
                                             50   DD 0008A          PUSHL   R0                                      ;
                                66           03   FB 0008C          CALLS   #3, LIB$SIGNAL                          ;
                                             04   0008F          RET                                             ; 2501
                                             50   D5 00090 7$:      TSTL    R0                                      ; 2529
                                             0D   13 00092          BEQL    8$                                      ;
                                             50   DD 00094          PUSHL   R0                                      ; 2535
                                             28   BB 00096          PUSHR   #^M<R3,R5>                              ;
                                             03   DD 00098          PUSHL   #3                                      ;
                                7E      1001 8F   3C 0009A          MOVZWL  #4097, -(SP)                            ;
                                             0B   11 0009F          BRB     9$                                      ;
                                             64   DD 000A1 8$:      PUSHL   RECORD_COUNT                            ; 2542
                                             28   BB 000A3          PUSHR   #^M<R3,R5>                              ;
                                             03   DD 000A5          PUSHL   #3                                      ;
                                7E      1009 8F   3C 000A7          MOVZWL  #4105, -(SP)                            ;
                        0000V   CF           01   FB 000AC 9$:      CALLS   #1, COPY$MSG_NUMBER                     ;
                                             50   DD 000B1          PUSHL   R0                                      ;
                                66           05   FB 000B3          CALLS   #5, LIB$SIGNAL                          ;
                                             04   000B6          RET                                             ; 2550

; Routine Size:  183 bytes,    Routine Base:  $CODE$ + 0811
```

```
;  2032          2551  1 ROUTINE REPORT_BYPASS (                                  ! Report the bypassing of an input file
;  2033          2552  1                         NUMBER )                         ! Error number
;  2034          2553  1                     : NOVALUE =
;  2035          2554  1
;  2036          2555  1 !++
;  2037          2556  1 ! FUNCTIONAL DESCRIPTION:
;  2038          2557  1 !
;  2039          2558  1 !       This routine reports the name of an input file which has been bypassed.
;  2040          2559  1 !
;  2041          2560  1 ! FORMAL PARAMETERS:
;  2042          2561  1 !
;  2043          2562  1 !       NUMBER.rlu.v - Error number
;  2044          2563  1 !
;  2045          2564  1 ! IMPLICIT INPUTS:
;  2046          2565  1 !
;  2047          2566  1 !       INFILE_NAM_BLK - Input file name block
;  2048          2567  1 !       INFILE_NAME - Input file resultant name
;  2049          2568  1 !       INFILE_XNAME - Input file expanded name
;  2050          2569  1 !
;  2051          2570  1 ! IMPLICIT OUTPUTS:
;  2052          2571  1 !
;  2053          2572  1 !       None
;  2054          2573  1 !
;  2055          2574  1 ! ROUTINE VALUE:
;  2056          2575  1 !
;  2057          2576  1 !       None
;  2058          2577  1 !
;  2059          2578  1 ! SIDE EFFECTS:
;  2060          2579  1 !
;  2061          2580  1 !       None
;  2062          2581  1 !
;  2063          2582  1 !--
;  2064          2583  1
;  2065          2584  2     BEGIN
;  2066          2585  2
;  2067          2586  2     LOCAL
;  2068          2587  2         NAME_DESC :  VECTOR[2];                         ! Input file name descriptor
;  2069          2588  2
;  2070          2589  2 !
;  2071          2590  2 ! Setup the input file name descriptor.
;  2072          2591  2 !
;  2073          2592  2
;  2074          2593  2     IF .INFILE_NAM_BLK[NAM$B_RSL] NEQ 0                    ! If RMS has setup a resultant name string,
;  2075          2594  2     THEN                                                   !
;  2076          2595  3         BEGIN                                              !
;  2077          2596  3         NAME_DESC[0] = .INFILE_NAM_BLK[NAM$B_RSL];         ! setup the name descriptor to use
;  2078          2597  3         NAME_DESC[1] = INFILE_NAME;                        ! the resultant name string.
;  2079          2598  3         END
;  2080          2599  2     ELSE
;  2081          2600  3         BEGIN                                              !
;  2082          2601  3         NAME_DESC[0] = .INFILE_NAM_BLK[NAM$B_ESL];         ! Otherwise, use the expanded name string.
;  2083          2602  3         NAME_DESC[1] = INFILE_XNAME;                       !
;  2084          2603  3         END;
;  2085          2604  2
;  2086          2605  2 !
;  2087          2606  2 ! Report the name of the input file which is being bypassed.
;  2088          2607  2 !
```

```
;  2089          2608  2
;  2090          2609  2      PUT_MESSAGEX( .NUMBER, 1, NAME_DESC );              ! Report the name of the input file.
;  2091          2610  2
;  2092          2611  2  !
;  2093          2612  2  ! Return to the caller.
;  2094          2613  2  !
;  2095          2614  2
;  2096          2615  2      RETURN;                                             ! Return to the caller.
;  2097          2616  2
;  2098          2617  1      END;


                                        0004 00000 REPORT_BYPASS:
                                                   .WORD   Save R2                                          ;  2551
                          52    0000V   CF  9E 00002         MOVAB   COPY$MSG_NUMBER, R2                    ;
                          5E            08  C2 00007         SUBL2   #8, SP                                 ;
                          50    0000G   CF  9A 0000A         MOVZBL  INFILE_NAM_BLK+3, R0                   ;  2593
                                        0B  13 0000F         BEQL    1$                                     ;
                          6E            50  D0 00011         MOVL    R0, NAME_DESC                          ;  2596
                    04    AE    0000G   CF  9E 00014         MOVAB   INFILE_NAME, NAME_DESC+4               ;  2597
                                        0B  11 0001A         BRB     2$                                     ;  2598
                          6E    0000G   CF  9A 0001C 1$:     MOVZBL  INFILE_NAM_BLK+11, NAME_DESC           ;  2601
                    04    AE    0000G   CF  9E 00021         MOVAB   INFILE_XNAME, NAME_DESC+4              ;  2602
                                    04  AC  DD 00027 2$:     PUSHL   NUMBER                                 ;  2609
                          62            01  FB 0002A         CALLS   #1, COPY$MSG_NUMBER                    ;
          7E        00                50      01  7A 0002D         EMUL    #1, R0, #0, -(SP)              ;
          50        50                8E      08  7B 00032         EDIV    #8, (SP)+, R0, R0              ;
                                      04      50  D1 00037         CMPL    R0, #4                         ;
                                              14  13 0003A         BEQL    3$                             ;
                                              5E  DD 0003C         PUSHL   SP                             ;
                                              01  DD 0003E         PUSHL   #1                             ;
                                        04    AC  DD 00040         PUSHL   NUMBER                         ;
                                      62      01  FB 00043         CALLS   #1, COPY$MSG_NUMBER            ;
                                              50  DD 00046         PUSHL   R0                             ;
                          00000000G   00      03  FB 00048         CALLS   #3, LIB$SIGNAL                 ;
                                              04  00004F         RET                                    ;
                                              5E  DD 00050 3$:     PUSHL   SP                             ;
                                              01  DD 00052         PUSHL   #1                             ;
                                        04    AC  DD 00054         PUSHL   NUMBER                         ;
                                      62      01  FB 00057         CALLS   #1, COPY$MSG_NUMBER            ;
                                              50  DD 0005A         PUSHL   R0                             ;
                          00000000G   00      03  FB 0005C         CALLS   #3, LIB$STOP                   ;
                                              04  00063         RET                                    ;  2617

; Routine Size:  100 bytes,    Routine Base:  $CODE$ + 08C8
```

```
;  2100          2618  1 GLOBAL ROUTINE COPY$LOG_MSG (                        ! Signal a COPY message
;  2101          2619  1                  NUMBER )                            ! Error number
;  2102          2620  1                  : NOVALUE =
;  2103          2621  1
;  2104          2622  1 !++
;  2105          2623  1 ! FUNCTIONAL DESCRIPTION:
;  2106          2624  1 !
;  2107          2625  1 !       This routine sends an informational message to the user if
;  2108          2626  1 !       activity reporting has been requested.
;  2109          2627  1 !
;  2110          2628  1 ! FORMAL PARAMETERS:
;  2111          2629  1 !
;  2112          2630  1 !       NUMBER.rlu.v - error number
;  2113          2631  1 !
;  2114          2632  1 ! IMPLICIT INPUTS:
;  2115          2633  1 !
;  2116          2634  1 !       LOG_MSG - Activity reporting indicator
;  2117          2635  1 !       OUTFILE_COUNT - Number of output files created
;  2118          2636  1 !       OUT_NAME_DESC - Output file name descriptor
;  2119          2637  1 !
;  2120          2638  1 ! IMPLICIT OUTPUTS:
;  2121          2639  1 !
;  2122          2640  1 !       None
;  2123          2641  1 !
;  2124          2642  1 ! ROUTINE VALUE:
;  2125          2643  1 !
;  2126          2644  1 !       None
;  2127          2645  1 !
;  2128          2646  1 ! SIDE EFFECTS:
;  2129          2647  1 !
;  2130          2648  1 !       None
;  2131          2649  1 !
;  2132          2650  1 !--
;  2133          2651  1
;  2134          2652  2     BEGIN
;  2135          2653  2
;  2136          2654  2 !
;  2137          2655  2 ! Return to the caller if activity reporting has not been requested.
;  2138          2656  2 !
;  2139          2657  2
;  2140          2658  2     IF NOT .LOG_MSG_QUAL                               ! If activity reporting is not requested,
;  2141          2659  2     THEN                                               !
;  2142          2660  2         RETURN;                                        ! return to the caller.
;  2143          2661  2
;  2144          2662  2 !
;  2145          2663  2 ! Call FAO to format the error message in the message buffer.
;  2146          2664  2 !
;  2147          2665  2
;  2148          2666  2     SELECTONE .NUMBER OF                               ! Select error message processing based
;  2149          2667  2         SET                                            ! on the actual error number.
;  2150          2668  2
;  2151          2669  2         [MSG$_NEWFILES]:
;  2152          2670  2             IF .OUTFILE_COUNT GEQU 2                   ! If at least 2 files was created,
;  2153          2671  2             THEN                                       !
;  2154        P 2672  2                 PUT_MESSAGE( MSG$_NEWFILES,            ! signal "<number> files created" with the following
;  2155        P 2673  2                              1,                        !    number of message arguments
;  2156          2674  2                              .OUTFILE_COUNT );         !    number of output files created
```

```
;  2157          2675  2
;  2158          2676  2          [MSG$_REPLACED, MSG$_OVERLAY, MSG$_CREATED]:
;  2159        P 2677  2              PUT_MESSAGEX( .NUMBER,                      ! signal the message with the following arguments:
;  2160        P 2678  2                            1,                             !   number of message arguments
;  2161          2679  2                            OUT_NAME_DESC );               !   address of the output name descriptor
;  2162          2680  2
;  2163          2681  2          [OTHERWISE]:
;  2164          2682  2              PUT_MESSAGEX( .NUMBER );                    ! Signal the appropriate message.
;  2165          2683  2          TES;
;  2166          2684  2
;  2167          2685  2      !
;  2168          2686  2      ! Return to the caller.
;  2169          2687  2      !
;  2170          2688  2
;  2171          2689  2      RETURN;                                             ! Return to the caller.
;  2172          2690  2
;  2173          2691  1      END;


                                                003C 00000          .ENTRY  COPY$LOG_MSG, Save R2,R3,R4,R5                  ; 2618
                                     55 00000000G   00   9E 00002   MOVAB   LIB$STOP, R5                                    ;
                                     54 00000000G   00   9E 00009   MOVAB   LIB$SIGNAL, R4                                  ;
                                     53      0000V  CF   9E 00010   MOVAB   COPY$MSG_NUMBER, R3                            ;
                          01         0000'  CF      01   E0 00015   BBS     #1, COPY$CLI_STATUS, 1$                         ; 2658
                                                04 0001B           RET                                                     ;
                                     52        04   AC   D0 0001C 1$: MOVL    NUMBER, R2                                  ; 2666
                                00001091       8F        52   D1 00020   CMPL    R2, #4241                               ; 2669
                                                         15   12 00027   BNEQ    3$                                      ;
                                     02        0000'  CF   D1 00029   CMPL    OUTFILE_COUNT, #2                       ; 2670
                                                         01   1E 0002E   BGEQU   2$                                      ;
                                                04 00030           RET                                                     ;
                                               0000'  CF   DD 00031 2$: PUSHL   OUTFILE_COUNT                           ; 2674
                                                         01   DD 00035   PUSHL   #1                                       ;
                                     7E        1091   8F   3C 00037   MOVZWL  #4241, -(SP)                               ;
                                                         37   11 0003C   BRB     5$                                      ;
                                00001073       8F        52   D1 0003E 3$: CMPL    R2, #4211                               ; 2676
                                                         12   13 00045   BEQL    4$                                      ;
                                000010AB       8F        52   D1 00047   CMPL    R2, #4267                               ;
                                                         09   13 0004E   BEQL    4$                                      ;
                                000010BB       8F        52   D1 00050   CMPL    R2, #4283                               ;
                                                         36   12 00057   BNEQ    7$                                      ;
                                                         52   DD 00059 4$: PUSHL   R2                                      ; 2679
                                     63                  01   FB 0005B   CALLS   #1, COPY$MSG_NUMBER                      ;
                          7E         00                  01   7A 0005E   EMUL    #1, R0, #0, -(SP)                       ;
                          50         50                  08   7B 00063   EDIV    #8, (SP)+, R0, R0                       ;
                                     04                  50   D1 00068   CMPL    R0, #4                                  ;
                                                         11   13 0006B   BEQL    6$                                      ;
                                               0000G  CF   9F 0006D   PUSHAB  OUT_NAME_DESC                              ;
                                                         01   DD 00071   PUSHL   #1                                       ;
                                                         52   DD 00073   PUSHL   R2                                       ;
                                     63                  01   FB 00075 5$: CALLS   #1, COPY$MSG_NUMBER                    ;
                                                         50   DD 00078   PUSHL   R0                                       ;
                                     64                  03   FB 0007A   CALLS   #3, LIB$SIGNAL                           ;
                                                         04 0007D           RET                                          ;
```

```
                                 0000G  CF  9F 0007E 68$:    PUSHAB  OUT_NAME_DESC                                    ;
                                        01  DD 00082         PUSHL   #1                                               ;
                                        52  DD 00084         PUSHL   R2                                               ;
                                    63  01  FB 00086         CALLS   #1, COPY$MSG_NUMBER                              ;
                                        50  DD 00089         PUSHL   R0                                               ;
                                    65  03  FB 0008B         CALLS   #3, LIB$STOP                                     ;
                                        04  0008E           RET                                                      ;
                                        52  DD 0008F 75$:    PUSHL   R2                                               ; 2682
                                    63  01  FB 00091         CALLS   #1, COPY$MSG_NUMBER                              ;
    7E          00                  50  01  7A 00094         EMUL    #1, R0, #0, -(SP)                                ;
    50          50                  8E  08  7B 00099         EDIV    #8, (SP)+, R0, R0                                ;
                                    04  50  D1 0009E         CMPL    R0, #4                                           ;
                                        0B  13 000A1         BEQL    8$                                               ;
                                        52  DD 000A3         PUSHL   R2                                               ;
                                    63  01  FB 000A5         CALLS   #1, COPY$MSG_NUMBER                              ;
                                        50  DD 000A8         PUSHL   R0                                               ;
                                    64  01  FB 000AA         CALLS   #1, LIB$SIGNAL                                   ;
                                        04  000AD           RET                                                      ;
                                        52  DD 000AE 80$:    PUSHL   R2                                               ;
                                    63  01  FB 000B0         CALLS   #1, COPY$MSG_NUMBER                              ;
                                        50  DD 000B3         PUSHL   R0                                               ;
                                    65  01  FB 000B5         CALLS   #1, LIB$STOP                                     ;
                                        04  000B8           RET                                                      ; 2691

; Routine Size:  185 bytes,    Routine Base:  $CODE$ + 092C
```

```
;  2175     2692  1 GLOBAL ROUTINE COPY$INOPN_ERR (                          ! RMS input open error action routine
;  2176     2693  1                  FAB_RAB_ADDRESS )                       ! Address of associated FAB or RAB
;  2177     2694  1                  : NOVALUE =
;  2178     2695  1
;  2179     2696  1 !++
;  2180     2697  1 ! FUNCTIONAL DESCRIPTION:
;  2181     2698  1 !
;  2182     2699  1 !       This RMS error action routine sends an input open error message to the user.
;  2183     2700  1 !
;  2184     2701  1 ! FORMAL PARAMETERS:
;  2185     2702  1 !
;  2186     2703  1 !       FAB_RAB_ADDRESS.ra.v - Address of the associated FAB or RAB
;  2187     2704  1 !
;  2188     2705  1 ! IMPLICIT INPUTS:
;  2189     2706  1 !
;  2190     2707  1 !       Input file name block
;  2191     2708  1 !       Input file name after open
;  2192     2709  1 !       Input file name before open
;  2193     2710  1 !       Input file cli descriptor
;  2194     2711  1 !
;  2195     2712  1 ! IMPLICIT OUTPUTS:
;  2196     2713  1 !
;  2197     2714  1 !       None
;  2198     2715  1 !
;  2199     2716  1 ! ROUTINE VALUE:
;  2200     2717  1 !
;  2201     2718  1 !       None
;  2202     2719  1 !
;  2203     2720  1 ! SIDE EFFECTS:
;  2204     2721  1 !
;  2205     2722  1 !       None
;  2206     2723  1 !
;  2207     2724  1 !--
;  2208     2725  1
;  2209     2726  2     BEGIN
;  2210     2727  2
;  2211     2728  2     BIND
;  2212     2729  2         FAB_RAB = .FAB_RAB_ADDRESS :  BLOCK[,BYTE];     ! Redefine routine parameter.
;  2213     2730  2
;  2214     2731  2     LOCAL
;  2215     2732  2         MESSAGE_ID,                                     ! Local message identifier
;  2216     2733  2         NAM_BLK : REF $BBLOCK[],                        ! Pointer to NAM block
;  2217     2734  2         NAME_DESC :  VECTOR[2];                         ! Input file name descriptor
;  2218     2735  2
;  2219     2736  2 !
;  2220     2737  2 ! Fillin the file name descriptor with the most complete name possible.
;  2221     2738  2 !
;  2222     2739  2
;  2223     2740  2     NAM_BLK = .FAB_RAB[FAB$L_NAM];
;  2224     2741  2
;  2225     2742  2     IF .NAM_BLK[NAM$B_RSL] NEQ 0                        ! If a resultant name string exists,
;  2226     2743  2     THEN                                                !
;  2227     2744  2         BEGIN                                           !
;  2228     2745  3         MESSAGE_ID = MSG$_OPENIN;                       ! indicate an open error
;  2229     2746  3         NAME_DESC[0] = .NAM_BLK[NAM$B_RSL];             ! and fillin the resultant name length
;  2230     2747  3         NAME_DESC[1] = .NAM_BLK[NAM$L_RSA];             ! and address.
;  2231     2748  3         END
```

```
;  2232          2749  2     ELSE
;  2233          2750  2         IF .NAM_BLK[NAM$B_ESL] NEQ 0                    ! If RMS created an expanded string
;  2234          2751  2         THEN                                            ! but coundn't open the file,
;  2235          2752  2             BEGIN                                       !
;  2236          2753  2             MESSAGE_ID = MSG$_OPENIN;                   ! indicate an open error
;  2237          2754  2             NAME_DESC[0] = .NAM_BLK[NAM$B_ESL];         ! and fillin the expanded name length
;  2238          2755  2             NAME_DESC[1] = .NAM_BLK[NAM$L_ESA];         ! and address.
;  2239          2756  2             END
;  2240          2757  2         ELSE
;  2241          2758  2             BEGIN
;  2242          2759  2             MESSAGE_ID = MSG$_OPENINX;                  ! Otherwise, indicate a fatal open error
;  2243          2760  2             NAME_DESC[0] = .INFILE_CLI_DESC[DSC$W_LENGTH];  ! and use the file name length
;  2244          2761  2             NAME_DESC[1] = .INFILE_CLI_DESC[DSC$A_POINTER]; ! and length passed by the CLI.
;  2245          2762  2             END;
;  2246          2763  2     !
;  2247          2764  2     ! If mag tape and operator aborted the mount, make it fatal
;  2248          2765  2     !
;  2249          2766  2         IF .FAB_RAB[$FAB_DEV(sdi)]
;  2250          2767  2             AND .FAB_RAB[FAB$L_STV] EQL SS$_ABORT
;  2251          2768  2         THEN
;  2252          2769  2             MESSAGE_ID = MSG$_OPENINX;
;  2253          2770  2
;  2254          2771  2     !
;  2255          2772  2     ! Signal the error condition.
;  2256          2773  2     !
;  2257          2774  2
;  2258        P 2775  2       PUT_MESSAGEX( .MESSAGE_ID,                        ! Signal "input open error" with the following argum
;  2259        P 2776  2                     1,                                  !    Number of message arguments
;  2260        P 2777  2                     NAME_DESC,                          !    Address of input name descriptor
;  2261        P 2778  2                     .FAB_RAB[FAB$L_STS],                !    Primary RMS completion code
;  2262          2779  2                     .FAB_RAB[FAB$L_STV] );              !    Secondary RMS completion code
;  2263          2780  2
;  2264          2781  2     !
;  2265          2782  2     ! Return to the caller.
;  2266          2783  2     !
;  2267          2784  2
;  2268          2785  2       RETURN;                                           ! Return to the caller.
;  2269          2786  2
;  2270          2787  1       END;


                                    001C 00000          .ENTRY  COPY$INOPN_ERR, Save R2,R3,R4                   ;  2692
                              54    0000V CF   9E 00002 MOVAB   COPY$MSG_NUMBER, R4                             ;
                              5E          08   C2 00007 SUBL2   #8, SP                                        ;
                              52      04  AC   D0 0000A MOVL    FAB_RAB_ADDRESS, R2                           ;  2729
                              50      28  A2   D0 0000E MOVL    40(R2), NAM_BLK                               ;  2740
                                      03  A0   95 00012 TSTB    3(NAM_BLK)                                    ;  2742
                                          10   13 00015 BEQL    1$                                            ;
                              53    109A  8F   3C 00017 MOVZWL  #4250, MESSAGE_ID                             ;  2745
                              6E      03  A0   9A 0001C MOVZBL  3(NAM_BLK), NAME_DESC                         ;  2746
                        04    AE      04  A0   D0 00020 MOVL    4(NAM_BLK), NAME_DESC+4                       ;  2747
                                          25   11 00025 BRB     3$                                            ;  2742
                                      0B  A0   95 00027 1$:     TSTB    11(NAM_BLK)                           ;  2750
                                          10   13 0002A BEQL    2$                                            ;
```

```
                                53      109A   8F  3C 0002C          MOVZWL  #4250, MESSAGE_ID                       ; 2753
                                6E        0B   A0  9A 00031          MOVZBL  11(NAM_BLK), NAME_DESC                  ; 2754
                          04    AE        0C   A0  D0 00035          MOVL    12(NAM_BLK), NAME_DESC+4                ; 2755
                                                 10  11 0003A          BRB     3$                                      ; 2750
                                53      109C   8F  3C 0003C 2$:       MOVZWL  #4252, MESSAGE_ID                       ; 2759
                                6E    0000G    CF  3C 00041          MOVZWL  INFILE_CLI_DESC, NAME_DESC              ; 2760
                          04    AE    0000G    CF  D0 00046          MOVL    INFILE_CLI_DESC+4, NAME_DESC+4          ; 2761
                    0B    40    A2          04  E1 0004C 3$:       BBC     #4, 64(R2), 4$                          ; 2766
                                2C        0C   A2  D1 00051          CMPL    12(R2), #44                             ; 2767
                                                 05  12 00055          BNEQ    4$                                      ; 2769
                                53      109C   8F  3C 00057          MOVZWL  #4252, MESSAGE_ID                       ; 2779
                                                 53  DD 0005C 4$:       PUSHL   MESSAGE_ID
                                64                   01  FB 0005E          CALLS   #1, COPY$MSG_NUMBER
                    7E    00    50                   01  7A 00061          EMUL    #1, R0, #0, -(SP)
                    50    50    8E                   08  7B 00066          EDIV    #8, (SP)+, R0, R0
                                04                   50  D1 0006B          CMPL    R0, #4
                                                     18  13 0006E          BEQL    5$
                                7E        08         A2  7D 00070          MOVQ    8(R2), -(SP)
                                          08         AE  9F 00074          PUSHAB  NAME_DESC
                                                     01  DD 00077          PUSHL   #1
                                                     53  DD 00079          PUSHL   MESSAGE_ID
                                64                   01  FB 0007B          CALLS   #1, COPY$MSG_NUMBER
                                                     50  DD 0007E          PUSHL   R0
                          00000000G   00             05  FB 00080          CALLS   #5, LIB$SIGNAL
                                                         04 00087          RET
                                7E        08         A2  7D 00088 5$:       MOVQ    8(R2), -(SP)
                                          08         AE  9F 0008C          PUSHAB  NAME_DESC
                                                     01  DD 0008F          PUSHL   #1
                                                     53  DD 00091          PUSHL   MESSAGE_ID
                                64                   01  FB 00093          CALLS   #1, COPY$MSG_NUMBER
                                                     50  DD 00096          PUSHL   R0
                          00000000G   00             05  FB 00098          CALLS   #5, LIB$STOP
                                                         04 0009F          RET                                     ; 2787

; Routine Size:  160 bytes,    Routine Base:  $CODE$ + 09E5
```

```
;  2272     2788  1 ROUTINE IN_READ_ERROR : NOVALUE =                    ! RMS input read error action routine
;  2273     2789  1
;  2274     2790  1 !++
;  2275     2791  1 ! FUNCTIONAL DESCRIPTION:
;  2276     2792  1 !
;  2277     2793  1 !       This RMS error action routine sends an input read error message to the user.
;  2278     2794  1 !
;  2279     2795  1 ! FORMAL PARAMETERS:
;  2280     2796  1 !
;  2281     2797  1 !       None
;  2282     2798  1 !
;  2283     2799  1 ! IMPLICIT INPUTS:
;  2284     2800  1 !
;  2285     2801  1 !       INFILE_RAB - Input file RAB
;  2286     2802  1 !       IN_NAME_DESC - Input file name descriptor
;  2287     2803  1 !
;  2288     2804  1 ! IMPLICIT OUTPUTS:
;  2289     2805  1 !
;  2290     2806  1 !       None
;  2291     2807  1 !
;  2292     2808  1 ! ROUTINE VALUE:
;  2293     2809  1 !
;  2294     2810  1 !       None
;  2295     2811  1 !
;  2296     2812  1 ! SIDE EFFECTS:
;  2297     2813  1 !
;  2298     2814  1 !       None
;  2299     2815  1 !
;  2300     2816  1 !--
;  2301     2817  1
;  2302     2818  2     BEGIN
;  2303     2819  2
;  2304     2820  2 !
;  2305     2821  2 ! Signal the input read error.
;  2306     2822  2 !
;  2307     2823  2
;  2308   P 2824  2     PUT_MESSAGE( MSG$_READERR,                         ! Signal a "read error" with the following arguments
;  2309   P 2825  2                  1,                                    !    Number of message arguments
;  2310   P 2826  2                  IN_NAME_DESC,                         !    Address of input file name descriptor
;  2311   P 2827  2                  .INFILE_RAB[RAB$L_STS],               !    Primary RMS completion code
;  2312     2828  2                  .INFILE_RAB[RAB$L_STV] );             !    Secondary RMS completion code
;  2313     2829  2
;  2314     2830  2 !
;  2315     2831  2 ! Return to the caller.
;  2316     2832  2 !
;  2317     2833  2
;  2318     2834  2     RETURN;                                            ! Return to the caller.
;  2319     2835  2
;  2320     2836  1     END;


                                0000 00000 IN_READ_ERROR:
                                           .WORD   Save nothing                                    ; 2788
                        7E      0000G  CF  7D 00002        MOVQ    INFILE_RAB+8, -(SP)          ; 2828
```

```
                                 0000G  CF  9F 00007         PUSHAB  IN_NAME_DESC                                ;
                                        01  DD 0000B         PUSHL   #1                                          ;
                              7E        10B2  8F  3C 0000D   MOVZWL  #4274, -(SP)                               ;
                      0000V   CF            01  FB 00012   CALLS   #1, COPY$MSG_NUMBER                         ;
                                            50  DD 00017   PUSHL   R0                                          ;
                  00000000G   00            05  FB 00019   CALLS   #5, LIB$SIGNAL                              ;
                                                04 00020   RET                                                 ; 2836

; Routine Size:  33 bytes,    Routine Base:  $CODE$ + 0A85
```

```
;  2322          2837  1 ROUTINE IN_CLOSE_ERROR (                                  ! RMS input close error action routine
;  2323          2838  1                         FAB_RAB_ADDRESS )                 ! Address of associated FAB or RAB
;  2324          2839  1                         : NOVALUE =
;  2325          2840  1
;  2326          2841  1 !++
;  2327          2842  1 ! FUNCTIONAL DESCRIPTION:
;  2328          2843  1 !
;  2329          2844  1 !       This RMS error action routine sends an input close error message to the user.
;  2330          2845  1 !
;  2331          2846  1 ! FORMAL PARAMETERS:
;  2332          2847  1 !
;  2333          2848  1 !       FAB_RAB_ADDRESS.ra.v - Address of the associated FAB or RAB
;  2334          2849  1 !
;  2335          2850  1 ! IMPLICIT INPUTS:
;  2336          2851  1 !
;  2337          2852  1 !       IN_NAME_DESC - Input file name descriptor
;  2338          2853  1 !
;  2339          2854  1 ! IMPLICIT OUTPUTS:
;  2340          2855  1 !
;  2341          2856  1 !       None
;  2342          2857  1 !
;  2343          2858  1 ! ROUTINE VALUE:
;  2344          2859  1 !
;  2345          2860  1 !       None
;  2346          2861  1 !
;  2347          2862  1 ! SIDE EFFECTS:
;  2348          2863  1 !
;  2349          2864  1 !       None
;  2350          2865  1 !
;  2351          2866  1 !--
;  2352          2867  1
;  2353          2868  2     BEGIN
;  2354          2869  2
;  2355          2870  2     BIND
;  2356          2871  2         FAB_RAB = .FAB_RAB_ADDRESS :  BLOCK[,BYTE];     ! Redefine routine parameter.
;  2357          2872  2
;  2358          2873  2 !
;  2359          2874  2 ! Signal an input close error.
;  2360          2875  2 !
;  2361          2876  2
;  2362        P 2877  2     PUT_MESSAGE( MSG$_CLOSEIN,                            ! Signal a "close error" with the following argument
;  2363        P 2878  2                  1,                                       !    Number of message arguments
;  2364        P 2879  2                  IN_NAME_DESC,                            !    Address of input file name descriptor
;  2365        P 2880  2                  .FAB_RAB[FAB$L_STS],                     !    Primary RMS completion code
;  2366          2881  2                  .FAB_RAB[FAB$L_STV] );                   !    Secondary RMS completion code
;  2367          2882  2
;  2368          2883  2 !
;  2369          2884  2 ! Return to the caller.
;  2370          2885  2 !
;  2371          2886  2
;  2372          2887  2     RETURN;                                               ! Return to the caller.
;  2373          2888  2
;  2374          2889  1     END;
```

```
                                        0000 00000 IN_CLOSE_ERROR:
                                                           .WORD   Save nothing                                    ; 2837
                         50      04     AC   D0 00002           MOVL    FAB_RAB_ADDRESS, R0                     ; 2871
                         7E      08     A0   7D 00006           MOVQ    8(R0), -(SP)                            ; 2881
                               0000G    CF   9F 0000A           PUSHAB  IN_NAME_DESC                            ;
                                        01   DD 0000E           PUSHL   #1                                      ;
                         7E    1052     8F   3C 00010           MOVZWL  #4178, -(SP)                            ;
                  0000V  CF              01   FB 00015           CALLS   #1, COPY$MSG_NUMBER                     ;
                                         50   DD 0001A           PUSHL   R0                                      ;
              00000000G  00              05   FB 0001C           CALLS   #5, LIB$SIGNAL                          ;
                                         04   00023           RET                                             ; 2889

; Routine Size:  36 bytes,    Routine Base:  $CODE$ + 0AA6
```

```
;  2376          2890  1 GLOBAL ROUTINE COPY$OUTOPN_ERR (                          ! RMS output open error action routine
;  2377          2891  1                  FAB_RAB_ADDRESS )                        ! Address of associated FAB or RAB
;  2378          2892  1                  : NOVALUE =
;  2379          2893  1
;  2380          2894  1 !++
;  2381          2895  1 ! FUNCTIONAL DESCRIPTION:
;  2382          2896  1 !
;  2383          2897  1 !       This RMS error action routine sends an output open error message to the user.
;  2384          2898  1 !
;  2385          2899  1 ! FORMAL PARAMETERS:
;  2386          2900  1 !
;  2387          2901  1 !       FAB_RAB_ADDRESS.ra.v - Address of the associated FAB or RAB
;  2388          2902  1 !
;  2389          2903  1 ! IMPLICIT INPUTS:
;  2390          2904  1 !
;  2391          2905  1 !       OUTFILE_NAM_BLK - Output file name block
;  2392          2906  1 !       OUTFILE_NAME - Output file name after open
;  2393          2907  1 !       OUTFILE_XNAME - Output file name before open
;  2394          2908  1 !       OUTFILE_DESC - Output file request descriptor
;  2395          2909  1 !
;  2396          2910  1 ! IMPLICIT OUTPUTS:
;  2397          2911  1 !
;  2398          2912  1 !       None
;  2399          2913  1 !
;  2400          2914  1 ! ROUTINE VALUE:
;  2401          2915  1 !
;  2402          2916  1 !       None
;  2403          2917  1 !
;  2404          2918  1 ! SIDE EFFECTS:
;  2405          2919  1 !
;  2406          2920  1 !       None
;  2407          2921  1 !
;  2408          2922  1 !--
;  2409          2923  1
;  2410          2924  2     BEGIN
;  2411          2925  2
;  2412          2926  2     BIND
;  2413          2927  2         FAB_RAB = .FAB_RAB_ADDRESS :  BLOCK[,BYTE];     ! Redefine routine parameter.
;  2414          2928  2
;  2415          2929  2     LOCAL
;  2416          2930  2         MESSAGE_ID,                                     ! Local message identifier
;  2417          2931  2         NAME_DESC :  VECTOR[2];                         ! Output file name descriptor
;  2418          2932  2
;  2419          2933  2 !
;  2420          2934  2 ! Fillin the file name descriptor with the most complete name possible.
;  2421          2935  2 !
;  2422          2936  2
;  2423          2937  2     IF .OUTFILE_NAM_BLK[NAMSB_RSL] NEQ 0                ! If a resultant name string exists,
;  2424          2938  2     THEN                                                !
;  2425          2939  2         BEGIN
;  2426          2940  2         MESSAGE_ID = MSG$_OPENOUT;                      ! indicate an open error
;  2427          2941  2         NAME_DESC[0] = .OUTFILE_NAM_BLK[NAMSB_RSL];     ! and fillin the resultant name length
;  2428          2942  2         NAME_DESC[1] = OUTFILE_NAME;                    ! and address.
;  2429          2943  2         END
;  2430          2944  2     ELSE
;  2431          2945  2         IF .OUTFILE_NAM_BLK[NAMSB_ESL] NEQ 0            ! If RMS created an expanded string but couldn't ope
;  2432          2946  2         THEN                                            !
```

```
  3            BEGIN                                         !
  3            MESSAGE_ID = MSG$_OPENOUT;                    ! indicate an open error
  3            NAME_DESC[0] = .OUTFILE_NAM_BLK[NAM$B_ESL];   ! and fillin the expanded name length
  3            NAME_DESC[1] = OUTFILE_XNAME;                 ! and address.
  3            END
  2        ELSE
  3            BEGIN
  3            MESSAGE_ID = MSG$_OPENOUTX;                   ! Otherwise, indicate a fatal open error
  3            NAME_DESC[0] = .OUT_NAME_DESC[ 0 ];           ! and use the file name length
  3            NAME_DESC[1] = .OUT_NAME_DESC[ 1 ]; ! and length passed by the CLI.
  3            END;
  2   !
  2   ! If mag tape and operator aborted the mount, make it fatal
  2   !
  2        IF .FAB_RAB[$FAB_DEV(sdi)]
  2            AND .FAB_RAB[FAB$L_STV] EQL SS$_ABORT
  2        THEN
  2            MESSAGE_ID = MSG$_OPENOUTX;
  2
  2   !
  2   ! Signal the error condition.
  2   !
  2
P 2      PUT_MESSAGEX( .MESSAGE_ID,                          ! Signal "output open error" with the following argu
P 2                  1,                                      !     Number of message arguments
P 2                  NAME_DESC,                              !     Address of output name descriptor
P 2                  .FAB_RAB[FAB$L_STS],                    !     Primary RMS completion code
  2                  .FAB_RAB[FAB$L_STV] );                  !     Secondary RMS completion code
  2
  2   !
  2   ! Return to the caller.
  2   !
  2
  2      RETURN;                                             ! Return to the caller.
  2
  1      END;
```

```
                         001C 00000        .ENTRY  COPY$OUTOPN_ERR, Save R2,R3,R4         ; 2890
        54      0000V  CF   9E 00002        MOVAB   COPY$MSG_NUMBER, R4                   ;
        5E             08   C2 00007        SUBL2   #8, SP                                ;
        52         04  AC   D0 0000A        MOVL    FAB_RAB_ADDRESS, R2                   ; 2927
        50      0000G  CF   9A 0000E        MOVZBL  OUTFILE_NAM_BLK+3, R0                 ; 2937
                       10   13 00013        BEQL    1$                                    ;
        53       10A2  8F   3C 00015        MOVZWL  #4258, MESSAGE_ID                     ; 2940
        6E             50   D0 0001A        MOVL    R0, NAME_DESC                         ; 2941
  04    AE      0000G  CF   9E 0001D        MOVAB   OUTFILE_NAME, NAME_DESC+4             ; 2942
                       21   11 00023        BRB     3$                                    ; 2937
        50      0000G  CF   9A 00025  1$:   MOVZBL  OUTFILE_NAM_BLK+11, R0                ; 2945
                       10   13 0002A        BEQL    2$                                    ;
        53       10A2  8F   3C 0002C        MOVZWL  #4258, MESSAGE_ID                     ; 2948
        6E             50   D0 00031        MOVL    R0, NAME_DESC                         ; 2949
  04    AE      0000G  CF   9E 00034        MOVAB   OUTFILE_XNAME, NAME_DESC+4            ; 2950
                       0A   11 0003A        BRB     3$                                    ; 2945
```

```
                                53      10A4    8F  3C 0003C 2$:    MOVZWL  #4260, MESSAGE_ID                       ; 2954
                                6E      0000G   CF  7D 00041        MOVQ    OUT_NAME_DESC, NAME_DESC                ; 2955
                    0B      40  A2              04  E1 00046 3$:    BBC     #4, 64(R2), 4$                          ; 2961
                                2C      0C      A2  D1 0004B        CMPL    12(R2), #44                             ; 2962
                                                05  12 0004F        BNEQ    4$
                                53      10A4    8F  3C 00051        MOVZWL  #4260, MESSAGE_ID                       ; 2964
                                                53  DD 00056 4$:    PUSHL   MESSAGE_ID                              ; 2974
                                64              01  FB 00058        CALLS   #1, COPY$MSG_NUMBER
                7E      00      50              01  7A 0005B        EMUL    #1, R0, #0, -(SP)
                50      50      8E              08  7B 00060        EDIV    #8, (SP)+, R0, R0
                                04              50  D1 00065        CMPL    R0, #4
                                                18  13 00068        BEQL    5$
                                7E      08      A2  7D 0006A        MOVQ    8(R2), -(SP)
                                        08      AE  9F 0006E        PUSHAB  NAME_DESC
                                                01  DD 00071        PUSHL   #1
                                                53  DD 00073        PUSHL   MESSAGE_ID
                                64              01  FB 00075        CALLS   #1, COPY$MSG_NUMBER
                                                50  DD 00078        PUSHL   R0
                        00000000G   00          05  FB 0007A        CALLS   #5, LIB$SIGNAL
                                                    04 00081        RET
                                7E      08      A2  7D 00082 5$:    MOVQ    8(R2), -(SP)
                                        08      AE  9F 00086        PUSHAB  NAME_DESC
                                                01  DD 00089        PUSHL   #1
                                                53  DD 0008B        PUSHL   MESSAGE_ID
                                64              01  FB 0008D        CALLS   #1, COPY$MSG_NUMBER
                                                50  DD 00090        PUSHL   R0
                        00000000G   00          05  FB 00092        CALLS   #5, LIB$STOP
                                                    04 00099        RET                                             ; 2982

; Routine Size:  154 bytes,    Routine Base:  $CODE$ + 0ACA
```

```
;  2470          2983  1 ROUTINE OUT_WRITE_ERROR : NOVALUE =                    ! RMS output write error action routine
;  2471          2984  1
;  2472          2985  1 !++
;  2473          2986  1 ! FUNCTIONAL DESCRIPTION:
;  2474          2987  1 !
;  2475          2988  1 !       This RMS error action routine sends an output read error message to the user.
;  2476          2989  1 !
;  2477          2990  1 ! FORMAL PARAMETERS:
;  2478          2991  1 !
;  2479          2992  1 !       None
;  2480          2993  1 !
;  2481          2994  1 ! IMPLICIT INPUTS:
;  2482          2995  1 !
;  2483          2996  1 !       OUTFILE_RAB - Output file RAB
;  2484          2997  1 !       OUT_NAME_DESC - Output file name descriptor
;  2485          2998  1 !
;  2486          2999  1 ! IMPLICIT OUTPUTS:
;  2487          3000  1 !
;  2488          3001  1 !       None
;  2489          3002  1 !
;  2490          3003  1 ! ROUTINE VALUE:
;  2491          3004  1 !
;  2492          3005  1 !       None
;  2493          3006  1 !
;  2494          3007  1 ! SIDE EFFECTS:
;  2495          3008  1 !
;  2496          3009  1 !       None
;  2497          3010  1 !
;  2498          3011  1 !--
;  2499          3012  1
;  2500          3013  2     BEGIN
;  2501          3014  2
;  2502          3015  2 !
;  2503          3016  2 ! Signal the output write error.
;  2504          3017  2 !
;  2505          3018  2
;  2506        P 3019  2     PUT_MESSAGE( MSG$_WRITEERR,                       ! Signal a "write error" with the following argument
;  2507        P 3020  2                  1,                                   !    Number of message arguments
;  2508        P 3021  2                  OUT_NAME_DESC,                       !    Address of output file name descriptor
;  2509        P 3022  2                  .OUTFILE_RAB[RAB$L_STS],             !    Primary RMS completion code
;  2510        P 3023  2                  .OUTFILE_RAB[RAB$L_STV] );           !    Secondary RMS completion code
;  2511          3024  2
;  2512          3025  2 !
;  2513          3026  2 ! Return to the caller.
;  2514          3027  2 !
;  2515          3028  2
;  2516          3029  2     RETURN;                                           ! Return to the caller.
;  2517          3030  2
;  2518          3031  1     END;


                                0000 00000 OUT_WRITE_ERROR:
                                                .WORD   Save nothing                                    ; 2983
                            7E      0000G   CF  7D 00002       MOVQ    OUTFILE_RAB*8, -(SP)            ; 3023
```

```
                                 0000G  CF  9F 00007         PUSHAB  OUT_NAME_DESC                                    ;
                                        01  DD 0000B         PUSHL   #1                                             ;
                                    7E  10D2  8F  3C 0000D   MOVZWL  #4306, -(SP)                                   ;
                              0000V CF        01  FB 00012   CALLS   #1, COPY$MSG_NUMBER                            ;
                                        50  DD 00017         PUSHL   R0                                             ;
                           00000000G  00        05  FB 00019   CALLS   #5, LIB$SIGNAL                              ;
                                        04 00020         RET                                                    ; 3031

; Routine Size:  33 bytes,    Routine Base:  $CODE$ + 0B64
```

```
   1 GLOBAL ROUTINE COPY$OCLOSE_ERR (                              ! RMS output close error action routine
   1                 FAB_RAB_ADDRESS )                             ! Address of associated FAB or RAB
   1                 : NOVALUE =
   1
   1 !++
   1 ! FUNCTIONAL DESCRIPTION:
   1 !
   1 !      This RMS error action routine sends an output close error message to the user.
   1 !
   1 ! FORMAL PARAMETERS:
   1 !
   1 !      FAB_RAB_ADDRESS.ra.v - Address of the associated FAB or RAB
   1 !
   1 ! IMPLICIT INPUTS:
   1 !
   1 !      OUT_NAME_DESC - Output file name descriptor
   1 !
   1 ! IMPLICIT OUTPUTS:
   1 !
   1 !      None
   1 !
   1 ! ROUTINE VALUE:
   1 !
   1 !      None
   1 !
   1 ! SIDE EFFECTS:
   1 !
   1 !      None
   1 !
   1 !--
   1
   2    BEGIN
   2
   2    BIND
   2        FAB_RAB = .FAB_RAB_ADDRESS :  BLOCK[,BYTE];     ! Redefine routine parameter.
   2
   2 !
   2 ! Signal an output close error.
   2 !
   2
P  2    PUT_MESSAGE( MSG$_CLOSEOUT,                            ! Signal a "close error" with the following argument
P  2                 1,                                        !    Number of message arguments
P  2                 OUT_NAME_DESC,                            !    Address of output file name descriptor
P  2                 .FAB_RAB[FAB$L_STS],                      !    Primary RMS completion code
   2                 .FAB_RAB[FAB$L_STV] );                    !    Secondary RMS completion code
   2
   2 !
   2 ! Return to the caller.
   2 !
   2
   2    RETURN;                                                ! Return to the caller.
   2
   1    END;
```

```
                                        0000 00000          .ENTRY  COPY$OCLOSE_ERR, Save nothing                   ; 3032
                                50      04   AC   D0 00002  MOVL    FAB_RAB_ADDRESS, R0                             ; 3066
                                7E      08   A0   7D 00006  MOVQ    8(R0), -(SP)                                    ; 3076
                                     0000G   CF   9F 0000A  PUSHAB  OUT_NAME_DESC                                   ;
                                             01   DD 0000E  PUSHL   #1                                             ;
                                7E      105A 8F   3C 00010  MOVZWL  #4186, -(SP)                                    ;
                          0000V CF              01   FB 00015  CALLS   #1, COPY$MSG_NUMBER                          ;
                                             50   DD 0001A  PUSHL   R0                                             ;
                      00000000G 00              05   FB 0001C  CALLS   #5, LIB$SIGNAL                              ;
                                             04   00023  RET                                                       ; 3084

; Routine Size:  36 bytes,    Routine Base:  $CODE$ + 0B85
```

```
;  2574          3085  1 GLOBAL ROUTINE COPY$MSG_NUMBER (                          ! COPY/APPEND message number generator
;  2575          3086  1                         MSG_ID ) =                        ! Message number
;  2576          3087  1
;  2577          3088  1 !++
;  2578          3089  1 ! FUNCTIONAL DESCRIPTION:
;  2579          3090  1 !
;  2580          3091  1 !       This routine return a COPY-specific or APPEND-specific message id
;  2581          3092  1 !       by inserting the appropriate facility identifier in the high word
;  2582          3093  1 !       of the message id which is passed by the caller.  This routine also
;  2583          3094  1 !       records the highest severity message encountered.
;  2584          3095  1 !
;  2585          3096  1 ! FORMAL PARAMETERS:
;  2586          3097  1 !
;  2587          3098  1 !       MSG_ID.rlu.v - Message id
;  2588          3099  1 !
;  2589          3100  1 ! IMPLICIT INPUTS:
;  2590          3101  1 !
;  2591          3102  1 !       APPEND_COMMAND = APPEND command indicator
;  2592          3103  1 !       MOST_SEVERE_ERR - Current most severe error id
;  2593          3104  1 !       OUTFILE_NAM_BLK - Output file name block - wildcard indicator
;  2594          3105  1 !
;  2595          3106  1 ! IMPLICIT OUTPUTS:
;  2596          3107  1 !
;  2597          3108  1 !       MOST_SEVERE_ERR - Most severe error id may be updated
;  2598          3109  1 !
;  2599          3110  1 ! ROUTINE VALUE:
;  2600          3111  1 !
;  2601          3112  1 !       Actual message id
;  2602          3113  1 !
;  2603          3114  1 ! SIDE EFFECTS:
;  2604          3115  1 !
;  2605          3116  1 !       None
;  2606          3117  1 !
;  2607          3118  1 !--
;  2608          3119  1
;  2609          3120  2     BEGIN
;  2610          3121  2
;  2611          3122  2     MAP                                                    !
;  2612          3123  2         MSG_ID :  BLOCK[,BYTE];                            ! Redefine the form of the input argument
;  2613          3124  2
;  2614          3125  2     LOCAL
;  2615          3126  2         ACTUAL_MSG_ID :  BLOCK[1];                         ! Actual message identifier
;  2616          3127  2
;  2617          3128  2 !
;  2618          3129  2 ! Calculate the actual message identifier.
;  2619          3130  2 !
;  2620          3131  2
;  2621          3132  2 IF .MSG_ID<16,16> EQL 0                                    ! If facility unspecified,
;  2622          3133  2 THEN
;  2623          3134  2     IF .APPEND_COMMAND                                     ! If this is an APPEND command,
;  2624          3135  2     THEN                                                   !
;  2625          3136  3         ACTUAL_MSG_ID = .MSG_ID + (APPEND_ID * 65536)      ! insert the APPEND facility code into the message i
;  2626          3137  2     ELSE                                                   ! If this is a COPY command,
;  2627          3138  3         ACTUAL_MSG_ID = .MSG_ID + (COPY_ID * 65536)        ! insert the COPY facility code into the message id.
;  2628          3139  2 ELSE
;  2629          3140  2     ACTUAL_MSG_ID = .MSG_ID;                               ! else use existing code
;  2630          3141  2
```

```
;  2631          3142  2 !
;  2632          3143  2 ! Update the "most severe error" if the current error is more severe.
;  2633          3144  2 !
;  2634          3145  2
;  2635          3146  2     IF NOT .ACTUAL_MSG_ID AND                          ! If the current message is not a success message an
;  2636          3147  2         (.MOST_SEVERE_ERR OR                           ! either this is the first error message
;  2637          3148  2         .ACTUAL_MSG_ID[STS$V_SEVERITY] GTRU            ! or the current message severity
;  2638          3149  2                 .MOST_SEVERE_ERR[STS$V_SEVERITY])      ! is greater than the previous severity,
;  2639          3150  2     THEN                                               !
;  2640          3151  2         MOST_SEVERE_ERR = .ACTUAL_MSG_ID OR            ! update the most severe message id
;  2641          3152  2                           STS$M_INHIB_MSG;             ! and turn on the "suppress message" indicator.
;  2642          3153  2
;  2643          3154  2 !
;  2644          3155  2 ! Return the actual message id to the caller.
;  2645          3156  2 !
;  2646          3157  2
;  2647          3158  2     RETURN .ACTUAL_MSG_ID;                             ! Return the actual message id to the caller.
;  2648          3159  2
;  2649          3160  1     END;


                                    0004 00000          .ENTRY  COPY$MSG_NUMBER, Save R2                        ; 3085
                          52  0000'  CF  9E 00002       MOVAB   MOST_SEVERE_ERR, R2                             ;
                              06    AC  B5 00007        TSTW    MSG_ID+2                                        ; 3132
                                    1A  12 0000A        BNEQ    2$                                              ;
                          0B      10  A2  E9 0000C     BLBC    COPY$CLI_STATUS, 1$                             ; 3134
                      50      04  AC 00710000  8F  C1 00010     ADDL3   #7405568, MSG_ID, ACTUAL_MSG_ID         ; 3136
                                    0F  11 00019        BRB     3$                                              ;
                      50      04  AC 00670000  8F  C1 0001B 1$: ADDL3   #6750208, MSG_ID, ACTUAL_MSG_ID         ; 3138
                                    04  11 00024        BRB     3$                                              ; 3134
                              50      04  AC  D0 00026 2$:      MOVL    MSG_ID, ACTUAL_MSG_ID                   ; 3140
                              17          50  E8 0002A 3$:      BLBS    ACTUAL_MSG_ID, 5$                       ; 3146
                              0C          62  E8 0002D         BLBS    MOST_SEVERE_ERR, 4$                     ; 3147
                  51          62          03  00  EF 00030     EXTZV   #0, #3, MOST_SEVERE_ERR, R1             ; 3149
                  51          50          03  00  ED 00035     CMPZV   #0, #3, ACTUAL_MSG_ID, R1               ;
                                          08  1B 0003A         BLEQU   5$                                      ;
                      62              50 10000000  8F  C9 0003C 4$:  BISL3   #268435456, ACTUAL_MSG_ID, MOST_SEVERE_ERR ; 3151
                                          04 00044 5$:          RET                                             ; 3160

; Routine Size:  69 bytes,    Routine Base:  $CODE$ + 0BA9
```

```
; 2651          3161  1 END
; 2652          3162  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  $GLOBAL$                       61  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                         92  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                       3054  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  COPY$COPY_FILE                180  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(9)
;  $OWN$                           4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                           Library Statistics
;                                       -------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1     9776       150         1       581         00:01.1
```



```
;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:COPYMAIN/OBJ=OBJ$:COPYMAIN MSRC$:COPYMAIN/UPDATE=(ENH$:COPYMAIN)

; Size:          3234 code + 157 data bytes
; Run Time:         01:05.8
; Elapsed Time:     02:27.9
; Lines/CPU Min:     2883
; Lexemes/CPU-Min: 23243
; Memory Used:  277 pages
; Compilation Complete
```

COPYMSG
REQ

COPYSPECS
LIS

COPY

COPY
MAP

VMSMAC
REQ

COPY
REQ

COPYCLI
LIS

COPYMAIN
LIS

COPYSEMAN
LIS